



# Inferring Vector Magnetic Fields from Stokes Profiles of GST/NIRIS Using a Convolutional Neural Network

Hao Liu<sup>1,2</sup>, Yan Xu<sup>1,3,4</sup>, Jiasheng Wang<sup>1,3,4</sup> , Ju Jing<sup>1,3,4</sup>, Chang Liu<sup>1,3,4</sup>, Jason T. L. Wang<sup>1,2</sup> , and Haimin Wang<sup>1,3,4</sup>

<sup>1</sup> Institute for Space Weather Sciences, New Jersey Institute of Technology, University Heights, Newark, NJ 07102-1982, USA; [hl422@njit.edu](mailto:hl422@njit.edu), [yan.xu@njit.edu](mailto:yan.xu@njit.edu), [jw438@njit.edu](mailto:jw438@njit.edu), [ju.jing@njit.edu](mailto:ju.jing@njit.edu), [chang.liu@njit.edu](mailto:chang.liu@njit.edu), [wangj@njit.edu](mailto:wangj@njit.edu), [haimin.wang@njit.edu](mailto:haimin.wang@njit.edu)

<sup>2</sup> Department of Computer Science, New Jersey Institute of Technology, University Heights, Newark, NJ 07102-1982, USA

<sup>3</sup> Center for Solar-Terrestrial Research, New Jersey Institute of Technology, University Heights, Newark, NJ 07102-1982, USA

<sup>4</sup> Big Bear Solar Observatory, New Jersey Institute of Technology, 40386 North Shore Lane, Big Bear City, CA 92314-9672, USA

Received 2019 December 5; revised 2020 March 29; accepted 2020 April 7; published 2020 May 7

## Abstract

We propose a new machine-learning approach to Stokes inversion based on a convolutional neural network (CNN) and the Milne–Eddington (ME) method. The Stokes measurements used in this study were taken by the Near InfraRed Imaging Spectropolarimeter (NIRIS) on the 1.6 m Goode Solar Telescope (GST) at the Big Bear Solar Observatory. By learning the latent patterns in the training data prepared by the physics-based ME tool, the proposed CNN method is able to infer vector magnetic fields from the Stokes profiles of GST/NIRIS. Experimental results show that our CNN method produces smoother and cleaner magnetic maps than the widely used ME method. Furthermore, the CNN method is four to six times faster than the ME method and able to produce vector magnetic fields in nearly real time, which is essential to space weather forecasting. Specifically, it takes  $\sim 50$  s for the CNN method to process an image of  $720 \times 720$  pixels comprising Stokes profiles of GST/NIRIS. Finally, the CNN-inferred results are highly correlated to the ME-calculated results and closer to the ME's results with the Pearson product-moment correlation coefficient (PPMCC) being closer to 1, on average, than those from other machine-learning algorithms, such as multiple support vector regression and multilayer perceptrons (MLP). In particular, the CNN method outperforms the current best machine-learning method (MLP) by 2.6%, on average, in PPMCC according to our experimental study. Thus, the proposed physics-assisted deep learning-based CNN tool can be considered as an alternative, efficient method for Stokes inversion for high-resolution polarimetric observations obtained by GST/NIRIS.

*Unified Astronomy Thesaurus concepts:* Solar magnetic fields (1503); Computational methods (1965); Convolutional neural networks (1938)

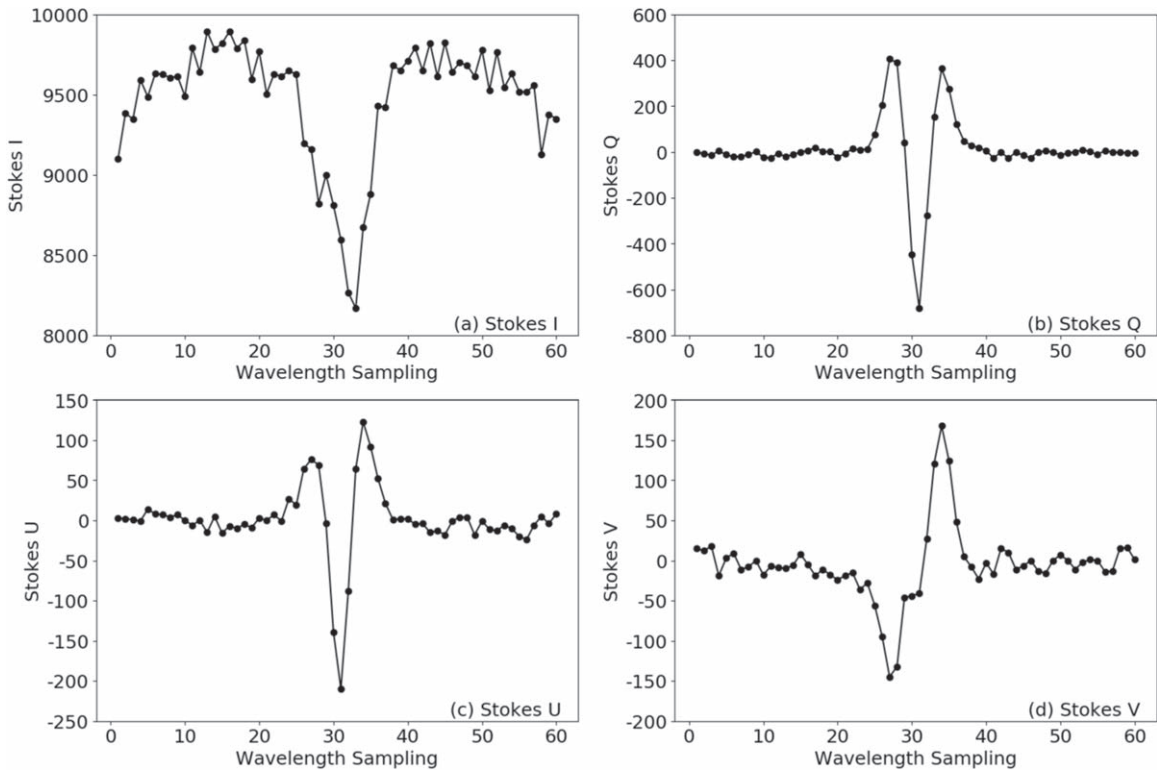
## 1. Introduction

Stokes inversion has been an important yet challenging task in solar physics for decades (Auer et al. 1977; del Toro Iniesta & Ruiz Cobo 1996; Asensio Ramos & de la Cruz Rodríguez 2015). Its purpose is to infer physical parameters, such as the total magnetic field strength, inclination and azimuth angles, Doppler shift of the line center, and so on, from spectropolarimetric data. In general, such an inversion task is accomplished by attempting to find an appropriate forward model that best describes the relationship between the spectral shapes of the four Stokes components and the physical parameters, which is essentially a nonlinear nonconvex inverse problem. In the past, several inversion models have been developed. Based on the Levenberg–Marquardt algorithm (Landolfi et al. 1984; Skumanich & Lites 1987; Press et al. 1991), a simplified model named the Milne–Eddington (ME) method (Auer et al. 1977; Landi Degl’Innocenti 1984) provides an analytical solution for fast evaluation of the required derivatives in the algorithm. Later, a more sophisticated method based on response functions was introduced by Ruiz Cobo & del Toro Iniesta (1992) that is able to retrieve height-dependent information. This method has several different implementations, including SPINOR (Frutiger et al. 2000), Helix+ (Lagg et al. 2004), and VFISV (Borrero et al. 2011).

In recent years, with rapid developments of advanced instruments and high-performance computers, powerful telescopes, such as the Daniel K. Inouye Solar Telescope (McMullin et al. 2012), European Solar Telescope (Collados 2008), and

Goode Solar Telescope (GST; Goode & Cao 2012) at the Big Bear Solar Observatory (BBSO), can produce data in unprecedented spatial and spectral resolution with high cadence. In order to process these data in a time that is practical on a human timescale, more efficient and stable automated methods are in demand. Many researchers have demonstrated that it is effective and efficient to perform Stokes inversion based on machine learning. For example, Socas-Navarro et al. (2001), Ruiz Cobo & Asensio Ramos (2012), and Quintero Noda et al. (2015) developed methods for transforming Stokes profiles to a low-dimensional space using principal component analysis, which reduces the computational load and makes subsequent inversions faster. Carroll & Staude (2001), Socas-Navarro (2003, 2005), and Carroll & Kopf (2008) employed multilayer perceptrons (MLP) for Stokes inversion, demonstrating the speed, noise tolerance, and stability of the MLP. Rees et al. (2004) and Teng (2015) used multiple support vector regression (MSVR) for real-time Stokes inversion. More recently, Asensio Ramos & Díaz Baso (2019) performed Stokes inversion based on convolutional neural networks (CNNs; LeCun et al. 2015) and applied their techniques to synthetic Stokes profiles obtained from snapshots of three-dimensional magnetohydrodynamic (MHD) numerical simulations of different structures of the solar atmosphere.

In this paper, we present a new machine-learning method, also based on CNNs, for Stokes inversion on the Near InfraRed Imaging Spectropolarimeter (NIRIS) data (Cao et al. 2012). Our CNN method differs from that of Asensio Ramos & Díaz Baso (2019) in two ways. First, Asensio Ramos & Díaz Baso (2019)



**Figure 1.** Stokes profiles of a pixel with an 857 G magnetic field strength,  $98^\circ$  inclination angle, and  $8^\circ$  azimuth angle calculated by the ME method. Each Stokes component has 60 wavelength sampling points.

used Stokes spectra synthesized in three-dimensional MHD simulations of the solar atmosphere and employed the CNNs to exploit all of the spatial information encoded in a training data set. In contrast, our method performs pixel-by-pixel inversions, exploiting the spatial information of the Stokes profiles in a pixel. Second, in the synthetic data used by Asensio Ramos & Díaz Baso (2019), each Stokes component has 112 spectral points. In contrast, in our NIRIS data, each Stokes component has 60 spectral points. Due to the different input sizes, the architecture of our CNN is different from those in Asensio Ramos & Díaz Baso (2019).

The rest of this paper is organized as follows. Section 2 describes the NIRIS data used in this study and our data collection scheme. Section 3 details our proposed CNN architecture and algorithm. Section 4 reports experimental results. Section 5 concludes the paper.

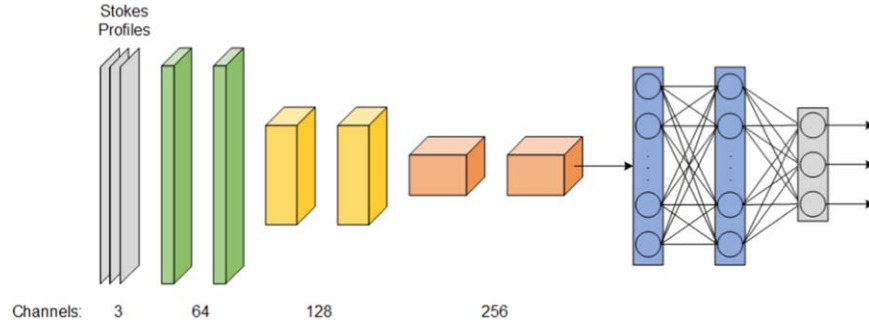
## 2. Data

The GST/NIRIS is the second generation of the InfraRed Imaging Magnetograph (Cao et al. 2006), offering unprecedented high-resolution vector magnetograms of the solar atmosphere from the deepest photosphere through the base of the corona. Its dual Fabry–Perot etalons provide an  $85''$  field of view with a cadence of 1 s for spectroscopic scans and 10 s for full Stokes measurements. The system utilizes half the chip to capture two simultaneous polarization states side by side and provides an image scale of  $0''.083 \text{ pixel}^{-1}$ . It produces full spectroscopic measurements  $I$ ,  $Q$ ,  $U$ , and  $V$  (Stokes profiles) at a spectral resolution of 0.01 nm in the FeI 1564.8 nm band, with a typical range of  $-0.25$  to  $+0.25$  nm from the line center (Wang et al. 2015, 2017; Xu et al. 2016, 2018; Liu et al. 2018). Figure 1 illustrates the Stokes  $I$ ,  $Q$ ,  $U$ , and  $V$  components of a

pixel with an 857 G magnetic field strength,  $98^\circ$  inclination angle, and  $8^\circ$  azimuth angle calculated by the ME method (Auer et al. 1977; Landi Degl’Innocenti 1984). Each Stokes component contains 60 wavelength sampling points.

We consider three active regions (ARs), namely, AR 12371, AR 12665, and AR 12673, on four different days. For AR 12371, we consider 10  $990 \times 950$  images collected at 10 different time points on 2015 June 22; we randomly select 1 million pixels (data samples) from these 10 images to form the training set. Then, again for AR 12371, we consider 10  $720 \times 720$  images collected at 10 different time points on 2015 June 25; we use the image collected at 20:00:00 UT on 2015 June 25 as the first test set. Next, we consider 10  $720 \times 720$  images from AR 12665 collected at 10 different time points on 2017 July 13; we use the image collected at 18:35:00 UT on 2017 July 13 as the second test set. Finally, we consider one  $720 \times 720$  image from AR 12673 collected at 19:18:00 UT on 2017 September 6 and use this image as the third test set. Each test set (image) has 518,400 pixels corresponding to 518,400 data samples. The training set and each of the test sets are disjoint. The first test set is of the same AR and within  $\sim 3$  days of the training set, while the second and third test sets are of different ARs just over 2 yr later. We want to see how well the trained CNN model works on these different test sets.

Each data sample (pixel) is comprised of Stokes  $I$ ,  $Q$ ,  $U$ , and  $V$  profiles taken at 60 spectral points. In addition, each data sample has a label, which is the vector magnetic field, including the total magnetic field strength; inclination angle; and azimuth angle, calculated by the ME method. During training, the labels of the data samples in the training set are used to train and optimize our CNN model. Because the labels of the training data are created by the physics-based ME



**Figure 2.** Architecture of our CNN. This network is comprised of an input layer, three convolutional blocks, two fully connected layers, and an output layer. The input of the CNN is a three-channel sequence of Stokes  $Q$ ,  $U$ , and  $V$  components each having 60 wavelength sampling points. The intermediate outputs of the three convolutional blocks have 64, 128, and 256 channels, respectively. There are 1024 neurons activated by ReLUs in both of the two fully connected layers. The output layer has three neurons activated by the Tanh function, where each neuron produces a value in the range  $(-1, 1)$  representing the total magnetic field strength, inclination angle, and azimuth angle, respectively.

method, our CNN model can be considered as a physics-assisted deep learning-based method.

During testing, we use the trained CNN model to predict or infer the label of a test data sample from the Stokes  $Q$ ,  $U$ , and  $V$  profiles, calibrated by the Stokes  $I$  component (Unno 1956), of the test data sample. We then compare the labels (i.e., vector magnetic fields) inferred by our CNN model with those calculated by the ME method for the test data samples under consideration. Because the Stokes profiles and labels have different units and scales, we normalize them as follows. For the Stokes profiles, we normalize them by dividing them by 1000. For the labels, we normalize the total magnetic field strength by dividing it by 5000 and the inclination and azimuth angles by dividing them by  $\pi$ . The two numbers, 1000 and 5000, are used here because most of the Stokes measurements have values between  $-1000$  and  $+1000$ , and their total magnetic field strengths range from  $-5000$  to  $+5000$  G.

After obtaining the estimated vector magnetic field, which is inferred by our trained model, of a test data sample (pixel), we can derive the three Cartesian components of the magnetic field, namely,  $B_x$ ,  $B_y$ , and  $B_z$ , of the pixel as follows:

$$\begin{cases} B_x = B_{\text{total}} \times \sin \phi \times \cos \theta \\ B_y = B_{\text{total}} \times \sin \phi \times \sin \theta, \\ B_z = B_{\text{total}} \times \cos \phi \end{cases} \quad (1)$$

where  $B_{\text{total}}$  denotes the total magnetic field strength,  $\phi$  is the inclination angle, and  $\theta$  is the azimuth angle.

### 3. Methodology

We use a CNN to infer vector magnetic fields from Stokes profiles of GST/NIRIS. Our CNN model helps in denoising inversions by exploiting the spatial information of the Stokes profiles. Figure 2 presents the architecture of our network. It contains an input layer, three convolutional blocks, two fully connected layers, and an output layer. The input layer receives a sequence of Stokes  $Q$ ,  $U$ , and  $V$  components, each having 60 wavelength sampling points, with three channels. Each channel corresponds to a Stokes component.

After the input layer, there are three convolutional blocks with the following structures. The first convolutional block consists of two convolutional layers that take, as input, the output from the previous layer and filter it with 64 kernels of sizes  $3 \times 1 \times 3$  and  $3 \times 1 \times 64$ , respectively, and a max-pooling layer with a pooling factor of 2. The second

convolutional block consists of two convolutional layers with filters of 128 kernels of sizes  $3 \times 1 \times 64$  and  $3 \times 1 \times 128$ , respectively, and a max-pooling layer with a pooling factor of 2. The third convolutional block consists of two convolutional layers with filters of 256 kernels of sizes  $3 \times 1 \times 128$  and  $3 \times 1 \times 256$ , respectively. The third convolutional block does not contain a max-pooling layer.

The activation functions used in both the convolutional and fully connected layers are rectified linear units (ReLUs; Goodfellow et al. 2016), defined as

$$\text{ReLU}(x) = \max(0, x) = \begin{cases} x & \text{if } x \geq 0 \\ 0 & \text{if } x < 0. \end{cases} \quad (2)$$

The output of the three convolutional blocks is flattened into a sequence, which is then sent to the two fully connected layers each having 1024 neurons activated by ReLUs. Finally, there is an output layer with three neurons activated by the hyperbolic tangent function (Tanh; Goodfellow et al. 2016), defined as

$$\text{Tanh}(x) = \left( \frac{e^x - e^{-x}}{e^x + e^{-x}} \right), \quad (3)$$

where each neuron outputs a value that lies in the range  $(-1, 1)$  representing the total magnetic field strength, inclination angle, and azimuth angle, respectively. The training of the CNN model is done by optimizing L1 loss, defined as follows (Goodfellow et al. 2016):

$$\text{L1 loss} = \frac{1}{N} \sum_{i=1}^N (|y_i^{\text{tot}} - \hat{y}_i^{\text{tot}}| + |y_i^{\text{inc}} - \hat{y}_i^{\text{inc}}| + |y_i^{\text{azi}} - \hat{y}_i^{\text{azi}}|), \quad (4)$$

where  $N = 1,000,000$  is the total number of pixels in the training set, and  $y_i^{\text{tot}}$ ,  $y_i^{\text{inc}}$ , and  $y_i^{\text{azi}}$  ( $\hat{y}_i^{\text{tot}}$ ,  $\hat{y}_i^{\text{inc}}$ , and  $\hat{y}_i^{\text{azi}}$ , respectively) denote the total magnetic field strength, inclination angle, and azimuth angle of the  $i$ th pixel calculated by the ME method (inferred by our CNN method), respectively. The L1 loss is chosen here because it is efficient and produces good results, as shown in Section 4.

Our CNN model is implemented in Python, TensorFlow, and Keras. A mini-batch strategy (LeCun et al. 2015; Goodfellow et al. 2016) is used to achieve faster convergence during back-propagation. The optimizer used is Adam (LeCun et al. 2015; Goodfellow et al. 2016), which is a stochastic gradient descent method. The initial learning rate is set to 0.001 with a learning

rate decay of 0.01 over each epoch,  $\beta_1$  is set to 0.9, and  $\beta_2$  is set to 0.999. The batch size is set to 256, and the number of epochs is set to 50.

During testing, to infer the physical parameters of each pixel in a test image, we take the Stokes  $Q$ ,  $U$ , and  $V$  profiles of the pixel and feed them to the trained CNN model. The CNN model will output a three-dimensional vector with normalized values in the range  $(-1, 1)$  representing the total magnetic field strength ( $B_{\text{total}}$ ), inclination angle ( $\phi$ ), and azimuth angle ( $\theta$ ), respectively. By denormalization of the values, we can obtain the inferred or estimated  $B_{\text{total}}$ ,  $\phi$ , and  $\theta$  of the pixel. Furthermore, based on the estimated  $B_{\text{total}}$ ,  $\phi$ , and  $\theta$ , we can derive the three Cartesian components of the magnetic field, namely,  $B_x$ ,  $B_y$ , and  $B_z$ , of the pixel using Equation (1).

## 4. Results

### 4.1. Performance Metrics

We conducted a series of experiments to evaluate the performance of the proposed CNN model and compare it with related methods based on four performance metrics: mean absolute error (MAE; Sen & Srivastava 1990), percent agreement (PA; McHugh 2012),  $R$ -squared (Sen & Srivastava 1990), and Pearson product-moment correlation coefficient (PPMCC; Galton 1886; Pearson 1895). We considered six quantities: total magnetic field strength ( $B_{\text{total}}$ ), inclination angle ( $\phi$ ), azimuth angle ( $\theta$ ),  $B_x$ ,  $B_y$ , and  $B_z$ . For each quantity, we compared its ME-calculated values with our CNN-inferred values and computed the four performance metrics.

The first performance metric is defined as (Sen & Srivastava 1990)

$$\text{MAE} = \frac{1}{N} \sum_{i=1}^N |y_i - \hat{y}_i|, \quad (5)$$

where  $N$  is the total number of data samples (pixels) in a test image, and  $y_i$  ( $\hat{y}_i$ ) denotes the ME-calculated (CNN-inferred) value for the  $i$ th pixel in the test image. This metric is used to quantitatively assess the dissimilarity (distance) between the ME-calculated and CNN-inferred values in the test image. The smaller the MAE is, the better performance a method has.

The second performance metric is defined as (McHugh 2012)

$$\text{PA} = \frac{M}{N} \times 100\%, \quad (6)$$

where  $M$  denotes the total number of agreement pixels in the test image. We say the  $i$ th pixel in the test image is an agreement pixel if  $|y_i - \hat{y}_i|$  is smaller than a user-specified threshold. (The default thresholds are set to 200 G for  $B_{\text{total}}$ ,  $B_x$ ,  $B_y$ , and  $B_z$  and  $10^\circ$  for  $\phi$  and  $\theta$ .) This metric is used to quantitatively assess the similarity between the ME-calculated and CNN-inferred values in the test image. The larger the PA is, the better performance a method has.

The third performance metric is defined as (Sen & Srivastava 1990)

$$R\text{-squared} = 1 - \frac{\sum_{i=1}^N (y_i - \hat{y}_i)^2}{\sum_{i=1}^N (y_i - \bar{y})^2}, \quad (7)$$

where  $\bar{y} = \frac{1}{N} \sum_{i=1}^N y_i$  denotes the mean of the ME-calculated values for all the pixels in the test image. The  $R$ -squared value, ranging from  $-\infty$  to 1, is used to measure the strength of the

relationship between the ME-calculated and CNN-inferred values in the test image. The larger (i.e., closer to 1) the  $R$ -squared value, the stronger the relationship between the ME-calculated and CNN-inferred values.

The fourth performance metric is defined as (Galton 1886; Pearson 1895)

$$\text{PPMCC} = \frac{E[(X - \mu_X)(Y - \mu_Y)]}{\sigma_X \sigma_Y}, \quad (8)$$

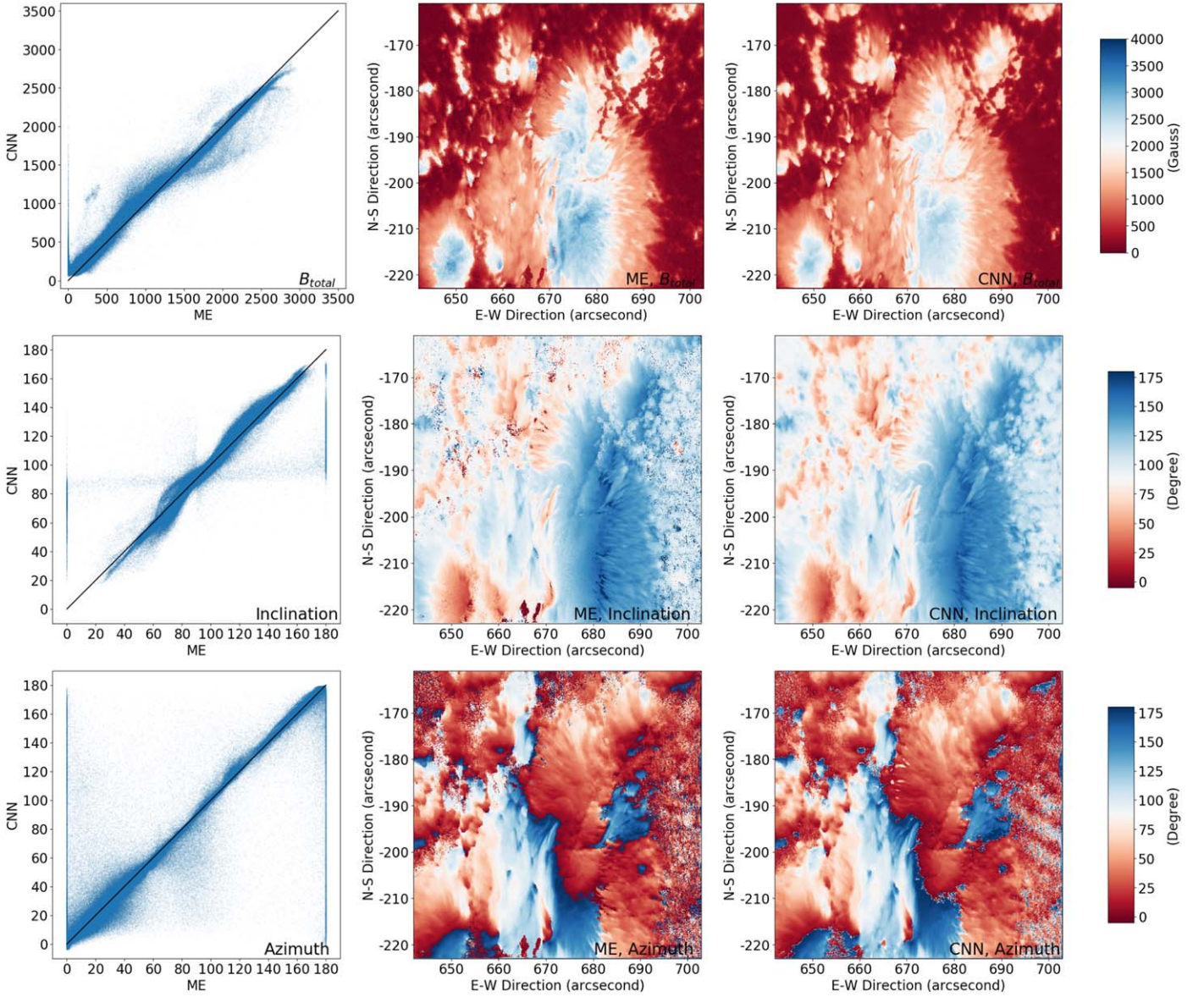
where  $X$  and  $Y$  represent the ME-calculated and CNN-inferred values, respectively;  $\mu_X$  and  $\mu_Y$  are the mean of  $X$  and  $Y$ , respectively;  $\sigma_X$  and  $\sigma_Y$  are the standard deviation of  $X$  and  $Y$ , respectively; and  $E(\cdot)$  is the expectation. The value of the PPMCC ranges from  $-1$  to  $1$ . A value of  $1$  means that a linear equation describes the relationship between  $X$  and  $Y$  perfectly where all data points are lying on a line for which  $Y$  increases as  $X$  increases. A value of  $-1$  means that all data points lie on a line for which  $Y$  decreases as  $X$  increases. A value of zero means that there is no linear correlation between the variables  $X$  and  $Y$ . We will mainly use the PPMCC in our experimental study because it measures the linear correlation between the ME-calculated and CNN-inferred values, quantifying how well the CNN-inferred values agree with the ME-calculated values in the test image (Galton 1886; Pearson 1895; Sen & Srivastava 1990). The larger (i.e., closer to 1) the PPMCC is, the better performance a method has. Notice that PA,  $R$ -squared, and PPMCC do not have units, while MAE has units: Gauss for  $B_{\text{total}}$ ,  $B_x$ ,  $B_y$ , and  $B_z$  and degrees for  $\phi$  (inclination angle) and  $\theta$  (azimuth angle).

### 4.2. Results of Using AR 12371 on 2015 June 22 as Training Data

In this experiment, we used the 1 million data samples (pixels) from AR 12371 collected on 2015 June 22 as the training data to train our CNN model. We then used the trained CNN model to infer vector magnetic fields from the Stokes  $Q$ ,  $U$ , and  $V$  profiles of the pixels in the three test sets (images) described in Section 2.<sup>5</sup> For comparison purposes, we also used the ME method (Auer et al. 1977; Landi Degl'Innocenti 1984) to derive the vector magnetic fields of the pixels in the three test images.

Figure 3 (Figures 4 and 5) presents results for the three obtained quantities  $B_{\text{total}}$ ,  $\phi$  (inclination angle), and  $\theta$  (azimuth angle), displayed from top to bottom in the figure, of the test image with  $720 \times 720$  pixels from AR 12371 (AR 12665 and AR 12673) collected on 2015 June 25 20:00:00 UT (2017 July 13 18:35:00 UT and 2017 September 6 19:18:00 UT). In all of the figures, the first column shows scatter plots for each obtained quantity. The  $X$ -axis and  $Y$ -axis in each scatter plot represent the values obtained by the ME and CNN methods, respectively. The black diagonal line in each scatter plot corresponds to pixels whose ME-calculated values are identical to CNN-inferred values. The second column shows magnetic maps with  $720 \times 720$  pixels derived by the ME method. The third column shows magnetic maps with  $720 \times 720$  pixels inferred by our CNN method.

<sup>5</sup> The source code and data sets used in the experiment can be downloaded from <https://web.njit.edu/~wangj/CNNStokesInversion/>.

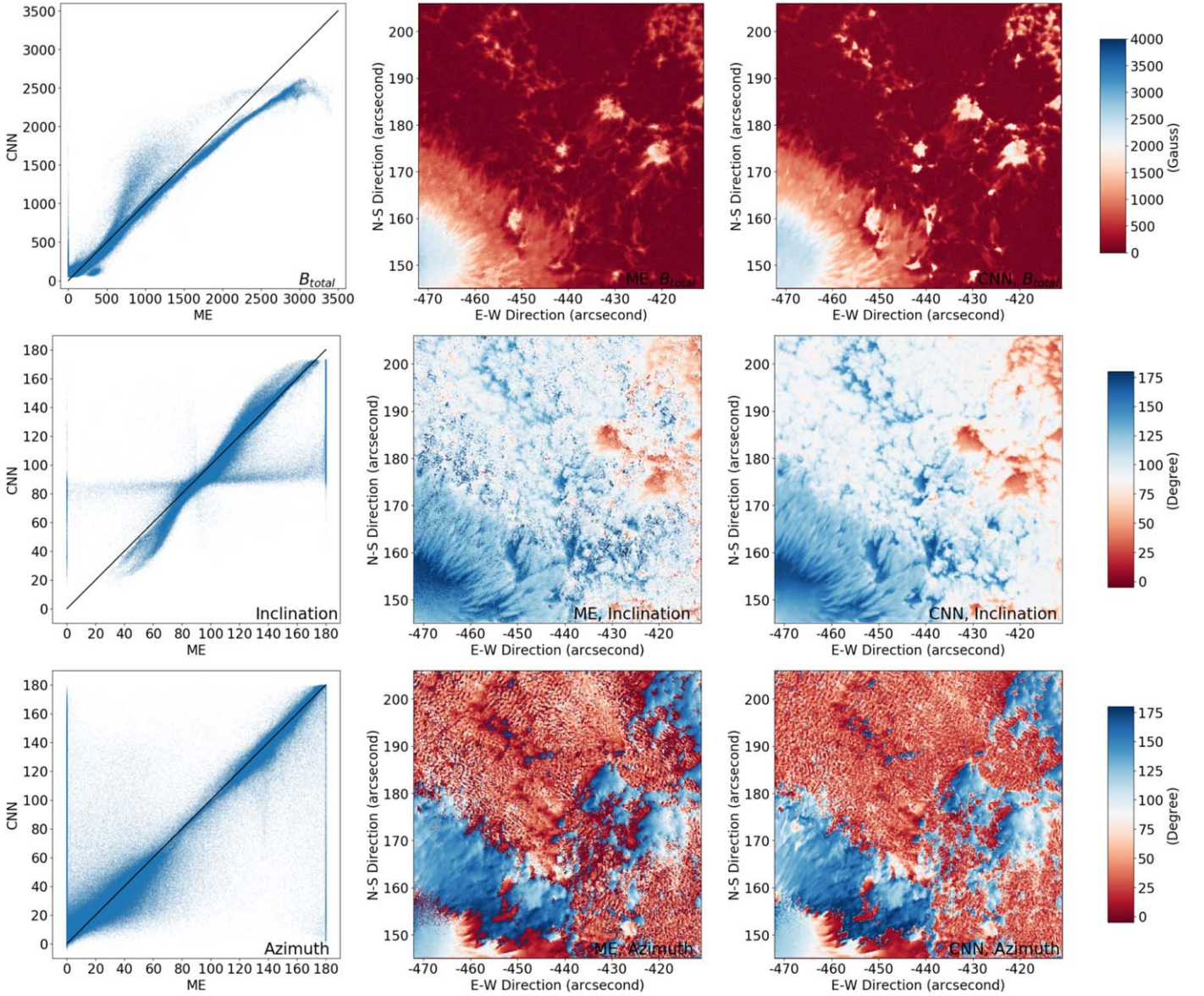


**Figure 3.** Comparison between the ME and CNN methods for deriving  $B_{\text{total}}$ ,  $\phi$  (inclination angle), and  $\theta$  (azimuth angle) based on the test image from AR 12371 collected on 2015 June 25 20:00:00 UT, where training data were taken from AR 12371 on 2015 June 22. Displayed from top to bottom are the results for  $B_{\text{total}}$ ,  $\phi$  (inclination angle), and  $\theta$  (azimuth angle). The first column shows scatter plots where the X-axis and Y-axis represent the values obtained by the ME and CNN methods, respectively. The black diagonal line in each scatter plot corresponds to pixels whose ME-calculated values are identical to CNN-inferred values. The second column shows magnetic maps derived by the ME method. The third column shows magnetic maps inferred by our CNN method.

*Summary of the results.* The scatter plots in the figures show that the Stokes inversion results obtained by our CNN method and the ME method are highly correlated. From the top left panels in Figures 3–5, we see that the CNN-inferred  $B_{\text{total}}$  values are closer to the ME-calculated  $B_{\text{total}}$  values in the low-field end and farther from the ME-calculated  $B_{\text{total}}$  values in the high-field end. The figures also show that the CNN method produces smoother and cleaner magnetic maps than the ME method. There are salt-and-pepper noise pixels in the magnetic maps produced by the ME method. To help locate the noise pixels, we use percentage difference images in which the value of the  $i$ th pixel is equal to  $(y_i - \hat{y}_i)/y_i \times 100\%$ , where  $y_i$  ( $\hat{y}_i$ ) denotes the ME-calculated (CNN-inferred) value for the  $i$ th pixel. For example, Figure 6 shows the percentage difference images for the  $\phi$  (inclination angle) maps in Figures 3–5. The percentage difference images highlight the locations of the

differences between the CNN-inferred  $\phi$  values and ME-calculated  $\phi$  values in the test images. Figure A1 (Figures A2 and A3) in the Appendix presents results for the quantities  $B_x$ ,  $B_y$ , and  $B_z$ , displayed from top to bottom in the figure, of the test image with  $720 \times 720$  pixels from AR 12371 (AR 12665 and AR 12673) collected on 2015 June 25 20:00:00 UT (2017 July 13 18:35:00 UT and 2017 September 6 19:18:00 UT).

To quantitatively assess the number of noise pixels in the magnetic maps derived by the ME and CNN methods, we adopt a threshold-based algorithm, which works as follows. We define  $P$  to be a noise pixel (outlier) with respect to a user-specified threshold if, among  $P$ 's eight neighboring pixels, there are more than four neighboring pixels satisfying the following condition: the difference between the value of a neighboring pixel and the value of  $P$  is greater than or equal to the threshold. The default thresholds are set to 500 G for  $B_{\text{total}}$ ,



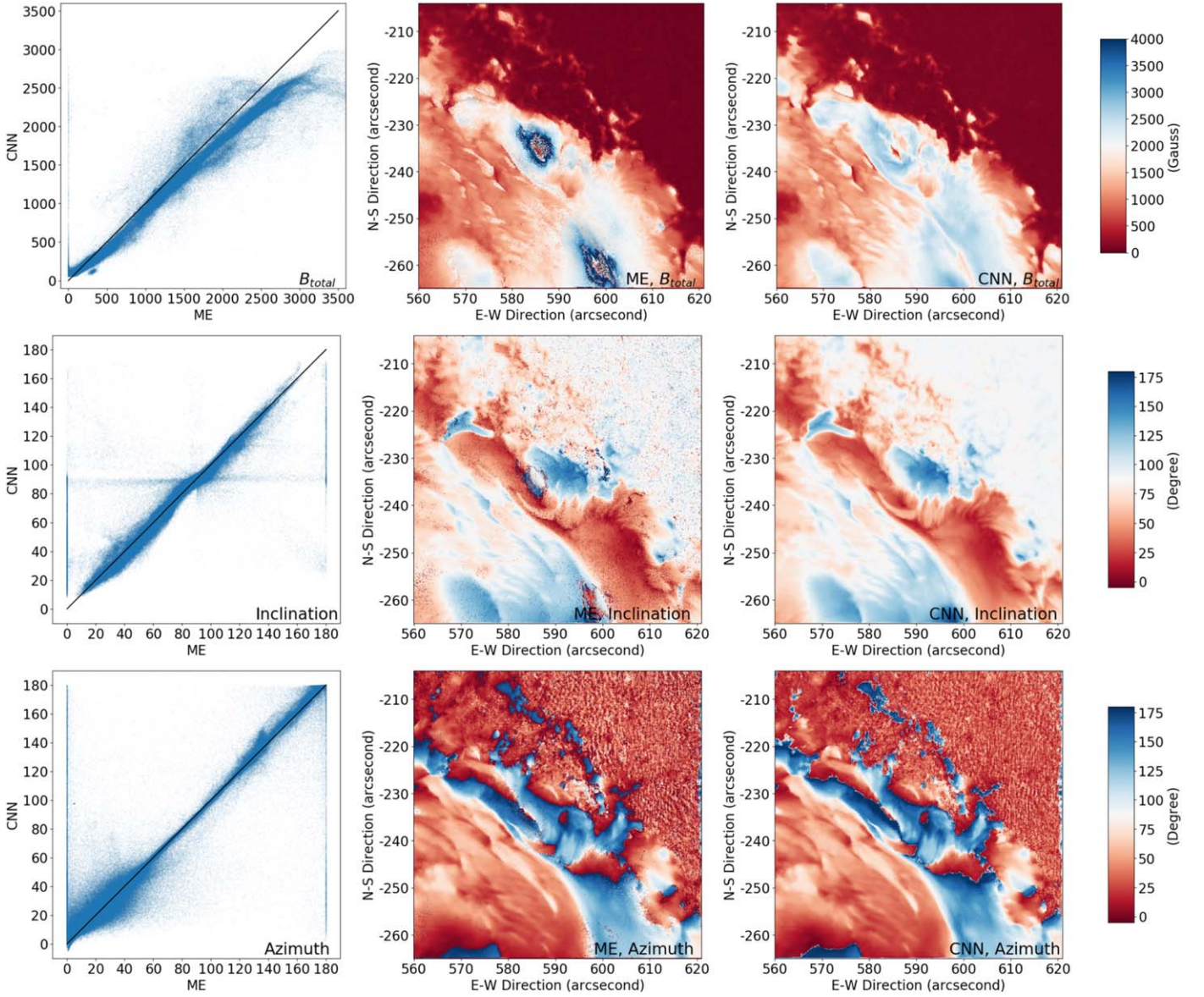
**Figure 4.** Comparison between the ME and CNN methods for deriving  $B_{\text{total}}$ ,  $\phi$  (inclination angle), and  $\theta$  (azimuth angle) based on the test image from AR 12665 collected on 2017 July 13 18:35:00 UT, where training data were taken from AR 12371 on 2015 June 22. Displayed from top to bottom are the results for  $B_{\text{total}}$ ,  $\phi$  (inclination angle), and  $\theta$  (azimuth angle). The first column shows scatter plots where the X-axis and Y-axis represent the values obtained by the ME and CNN methods, respectively. The black diagonal line in each scatter plot corresponds to pixels whose ME-calculated values are identical to CNN-inferred values. The second column shows magnetic maps derived by the ME method. The third column shows magnetic maps inferred by our CNN method.

$B_x$ ,  $B_y$ , and  $B_z$  and  $20^\circ$  for  $\phi$  (inclination angle) and  $\theta$  (azimuth angle). We define the outlier difference to be the number of outliers produced by the ME method minus the number of outliers produced by our CNN method. A positive outlier difference means ME produces more outliers than CNN, while a negative outlier difference means CNN produces more outliers than ME.

Table 1 presents the performance metric values of the CNN method. The results in Table 1 are consistent with those in Figures 3–A3. Specifically, the CNN-inferred results are highly correlated to the ME-calculated results, with PPMCC values being close to 1. Furthermore, CNN produces smoother magnetic maps with fewer outliers (noise pixels) than the ME method. This happens because, among the 1 million training data samples whose labels are calculated by the ME method, there are relatively few outliers. The CNN method can learn

latent patterns from the majority of the training data samples, which are clean. As a consequence, we obtain a good CNN model capable of producing clean results. Tables A1 and A2 in the Appendix present the performance metric values for the test images from AR 12371 and AR 12665 collected at 10 different time points on 2015 June 25 and 2017 July 13, respectively. The results in these tables are consistent with those in Table 1.

*Comparison with related methods.* To further understand the behavior of our CNN method and compare it with related machine-learning algorithms, we conduct a cross-validation study as follows. We partition the training set of 1 million data samples from AR 12371 on 2015 June 22 into 10 equal-sized folds. For every two training folds  $i$  and  $j$ ,  $i \neq j$ , folds  $i$  and  $j$  are disjoint. The first test set contains the 10  $720 \times 720$  images, also from AR 12371, collected on 2015 June 25. These test images are numbered from 1 to 10. In run  $i$ ,  $1 \leq i \leq 10$ , all

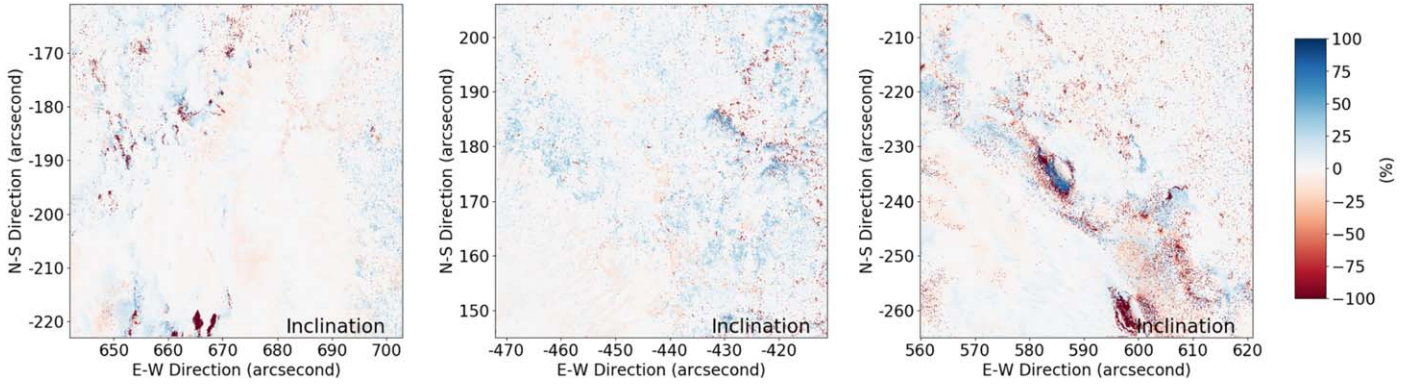


**Figure 5.** Comparison between the ME and CNN methods for deriving  $B_{\text{total}}$ ,  $\phi$  (inclination angle), and  $\theta$  (azimuth angle) based on the test image from AR 12673 collected on 2017 September 6 19:18:00 UT, where training data were taken from AR 12371 on 2015 June 22. Displayed from top to bottom are the results for  $B_{\text{total}}$ ,  $\phi$  (inclination angle), and  $\theta$  (azimuth angle). The first column shows scatter plots where the X-axis and Y-axis represent the values obtained by the ME and CNN methods, respectively. The black diagonal line in each scatter plot corresponds to pixels whose ME-calculated values are identical to CNN-inferred values. The second column shows magnetic maps derived by the ME method. The third column shows magnetic maps inferred by our CNN method.

training data samples except those in training fold  $i$  are used to train a machine-learning model, and the trained model is then used to make predictions on test image  $i$ . We calculate the performance metrics MAE, PA,  $R$ -squared, PPMCC, and outlier difference based on the predictions made in run  $i$ . There are 10 runs. The means and standard deviations over the 10 runs are calculated and recorded. We also conduct the same cross-validation study for the second test set containing the 10  $720 \times 720$  images from AR 12665 collected on 2017 July 13 and the third test set containing the  $720 \times 720$  image from AR 12673 collected on 2017 September 6. The third test set has only one image; hence, in each run, the same test image is used.

The related machine-learning algorithms considered here include MSVR (Rees et al. 2004; Teng 2015) and MLP (Carroll & Staude 2001; Socas-Navarro 2003, 2005; Carroll & Kopf 2008). The MSVR method uses the radial basis function

kernel. The MLP model consists of an input layer, an output layer, and two hidden layers, both with 1024 neurons. Table 2 (Tables 3 and 4) presents the mean MAE, PA,  $R$ -squared, PPMCC, outlier difference, and standard deviation for each quantity  $B_{\text{total}}$ ,  $B_x$ ,  $B_y$ ,  $B_z$ ,  $\phi$  (inclination angle), and  $\theta$  (azimuth angle) inferred by each of the three machine-learning methods, MSVR, MLP, and our CNN, for the first (second and third) test set. In the tables, PA,  $R$ -squared, PPMCC, and outlier difference do not have units, while MAE has units: Gauss for  $B_{\text{total}}$ ,  $B_x$ ,  $B_y$ , and  $B_z$  and degrees for  $\phi$  (inclination angle) and  $\theta$  (azimuth angle) respectively. It can be seen from the tables that the CNN-inferred results are highly correlated to the ME-calculated results and closer to the ME's results with PPMCC values being closer to 1, on average, than those from the other two machine-learning methods. In particular, based on the calculations on the six quantities  $B_{\text{total}}$ ,  $B_x$ ,  $B_y$ ,  $B_z$ ,  $\phi$  (inclination



**Figure 6.** Percentage difference images for the  $\phi$  (inclination angle) maps. The first panel shows the percentage difference image based on the test image from AR 12371 collected on 2015 June 25 20:00:00 UT. The second panel shows the percentage difference image based on the test image from AR 12665 collected on 2017 July 13 18:35:00 UT. The third panel shows the percentage difference image based on the test image from AR 12673 collected on 2017 September 6 19:18:00 UT. These percentage difference images highlight the locations of the differences between the CNN-inferred  $\phi$  values and ME-calculated  $\phi$  values in the three test images.

**Table 1**  
Performance Metric Values of Our CNN Method Based on the Test Images from Three ARs<sup>a</sup>

|                                    |                                 | $B_{\text{total}}$ | $B_x$   | $B_y$   | $B_z$   | $\phi$ | $\theta$ |
|------------------------------------|---------------------------------|--------------------|---------|---------|---------|--------|----------|
| 2015 Jun 25 20:00:00 UT (AR 12371) | MAE                             | 86.660             | 88.997  | 66.140  | 55.653  | 4.867  | 11.136   |
|                                    | PA                              | 91.6%              | 91.3%   | 95.2%   | 94.7%   | 92.2%  | 79.1%    |
|                                    | $R$ -squared                    | 0.963              | 0.936   | 0.901   | 0.976   | 0.838  | 0.720    |
|                                    | PPMCC                           | 0.983              | 0.968   | 0.951   | 0.989   | 0.916  | 0.853    |
|                                    | Outlier difference <sup>b</sup> | 2959               | 4380    | -770    | 1050    | 15108  | 7219     |
| 2017 Jul 13 18:35:00 UT (AR 12665) | MAE                             | 73.684             | 71.555  | 51.170  | 49.023  | 7.573  | 17.437   |
|                                    | PA                              | 91.5%              | 93.3%   | 96.4%   | 92.6%   | 84.8%  | 60.6%    |
|                                    | $R$ -squared                    | 0.950              | 0.841   | 0.851   | 0.941   | 0.663  | 0.665    |
|                                    | PPMCC                           | 0.976              | 0.918   | 0.926   | 0.971   | 0.827  | 0.821    |
|                                    | Outlier difference              | 3801               | 7280    | 3413    | 2478    | 35649  | 28274    |
| 2017 Sep 6 19:18:00 UT (AR 12673)  | MAE                             | 193.680            | 146.100 | 124.783 | 136.892 | 5.497  | 9.009    |
|                                    | PA                              | 75.0%              | 80.1%   | 86.2%   | 87.2%   | 91.3%  | 79.1%    |
|                                    | $R$ -squared                    | 0.841              | 0.884   | 0.777   | 0.736   | 0.776  | 0.807    |
|                                    | PPMCC                           | 0.935              | 0.943   | 0.888   | 0.859   | 0.881  | 0.902    |
|                                    | Outlier difference              | 19651              | 22317   | 16592   | 12950   | 21951  | 14265    |

#### Notes.

<sup>a</sup> The performance metric values in the table are obtained by training the CNN model using 1 million pixels from AR 12371 collected on 2015 June 22 and then applying the trained model to the test image from AR 12371 collected on 2015 June 25 20:00:00 UT (AR 12665 collected on 2017 July 13 18:35:00 UT and AR 12673 collected on 2017 September 6 19:18:00 UT).

<sup>b</sup> A positive outlier difference means ME produces more outliers than CNN, while a negative outlier difference means CNN produces more outliers than ME.

angle), and  $\theta$  (azimuth angle) in Tables 2–4, our CNN method outperforms the current best machine-learning method (MLP) by 2.6%, on average, in PPMCC. However, there is no definite conclusion about outlier differences among the three machine-learning methods.

#### 4.3. Results of Using Different ARs as Training Data

In the previous subsection, we use data points (pixels) from AR 12371 on 2015 June 22 as training data. In this subsection, we conduct additional experiments by varying the training data as follows. There are four data sets,  $D_1$ ,  $D_2$ ,  $D_3$ , and  $D_4$ , containing the images from AR 12371 on 2015 June 22, AR 12371 on 2015 June 25, AR 12665 on 2017 July 13, and AR 12673 on 2017 September 6, respectively. In each experiment, we randomly select 1 million pixels (data samples) from one or more data sets to form a training set. The CNN model is trained on this training set, and the trained model is then used to perform Stokes inversion on a test image. This test image must

be from a data set that is different from those data sets used to construct the training set. The time points for the test image are 17:33:00 UT on 2015 June 22, 20:00:00 UT on 2015 June 25, 18:35:00 UT on 2017 July 13, and 19:18:00 UT on 2017 September 6. We use  $D_x^{\text{train}} \rightarrow D_w^{\text{test}}$  ( $D_{x,y}^{\text{train}} \rightarrow D_w^{\text{test}}$  and  $D_{x,y,z}^{\text{train}} \rightarrow D_w^{\text{test}}$ ) to represent the experiment that uses training data samples from  $D_x$  ( $D_x$  and  $D_y$  and  $D_x$ ,  $D_y$ , and  $D_z$ ) and test data samples (pixels) from  $D_w$ , where  $1 \leq x, y, z, w \leq 4$ . Because  $D_4$  has only one  $720 \times 720$  image with 518,400 pixels,  $D_4$  alone is not used as a training set. Hence, there are 25 experiments in total. In each experiment, we calculate the performance metrics MAE, PA,  $R$ -squared, PPMCC, and outlier difference. Tables A3–A6 in the Appendix present the experimental results. Major findings based on these tables are summarized below.

1. Our CNN-inferred and ME-calculated results are highly correlated and close to each other with a PPMCC of  $\sim 0.9$

**Table 2**  
Performance Metric Values of MSVR, MLP, and Our CNN Method Based on the Test Set from AR 12371 Collected on 2015 June 25 <sup>a,b</sup>

|                                 |      | $B_{\text{total}}$ | $B_x$              | $B_y$              | $B_z$              | $\phi$             | $\theta$           |
|---------------------------------|------|--------------------|--------------------|--------------------|--------------------|--------------------|--------------------|
| MAE                             | MSVR | 437.02 (27.44)     | 712.02 (19.03)     | 706.51 (12.24)     | 339.26 (17.86)     | 23.02 (0.73)       | 84.43 (1.53)       |
|                                 | MLP  | 115.68 (5.15)      | 109.44 (7.60)      | 86.08 (4.80)       | 80.62 (4.19)       | 5.85 (0.31)        | 12.29 (1.72)       |
|                                 | CNN  | 81.57 (3.66)       | 76.56 (5.63)       | 58.83 (2.86)       | 52.18 (2.22)       | 4.54 (0.23)        | 9.34 (1.04)        |
| PA                              | MSVR | 34.7% (0.5%)       | 48.4% (1.0%)       | 44.6% (1.1%)       | 15.2% (1.5%)       | 5.6% (0.2%)        | 4.1% (0.5%)        |
|                                 | MLP  | 86.2% (0.7%)       | 88.4% (0.8%)       | 91.5% (0.5%)       | 89.5% (0.7%)       | 89.4% (1.0%)       | 76.7% (1.0%)       |
|                                 | CNN  | 91.6% (0.7%)       | 92.5% (0.8%)       | 96.1% (0.5%)       | 95.1% (0.4%)       | 93.6% (0.6%)       | 81.4% (1.4%)       |
| R-squared                       | MSVR | 0.45 (0.05)        | −0.92 (0.07)       | −5.34 (0.37)       | 0.28 (0.08)        | −0.09 (0.03)       | −2.80 (0.16)       |
|                                 | MLP  | 0.92 (0.01)        | 0.91 (0.01)        | 0.85 (0.01)        | 0.93 (0.01)        | 0.80 (0.01)        | 0.73 (0.04)        |
|                                 | CNN  | 0.97 (0.01)        | 0.94 (0.01)        | 0.93 (0.01)        | 0.97 (0.01)        | 0.83 (0.01)        | 0.76 (0.03)        |
| PPMCC <sup>c</sup>              | MSVR | 0.82 (0.01)        | −0.09 (0.04)       | −0.10 (0.08)       | 0.89 (0.01)        | 0.85 (0.01)        | 0.48 (0.03)        |
|                                 | MLP  | 0.97 (0.01)        | 0.96 (0.01)        | 0.93 (0.01)        | 0.97 (0.01)        | 0.91 (0.01)        | 0.86 (0.02)        |
|                                 | CNN  | <b>0.98</b> (0.01) | <b>0.97</b> (0.01) | <b>0.96</b> (0.01) | <b>0.99</b> (0.01) | <b>0.91</b> (0.01) | <b>0.88</b> (0.02) |
| Outlier difference <sup>d</sup> | MSVR | 2572 (945)         | 3009 (511)         | −1794 (555)        | −2038 (688)        | 13864 (887)        | 38828 (2083)       |
|                                 | MLP  | 3056 (823)         | 3587 (679)         | −208 (166)         | 1417 (403)         | 14495 (877)        | 13526 (2480)       |
|                                 | CNN  | 3060 (809)         | 3415 (488)         | −419 (235)         | 1436 (380)         | 14503 (883)        | 12645 (2930)       |

**Notes.**<sup>a</sup> Each number in the table represents the average value of 10 experiments.<sup>b</sup> Standard deviations are enclosed in parentheses.<sup>c</sup> The best PPMCC values achieved by the three machine-learning methods are highlighted in bold.<sup>d</sup> A positive outlier difference means ME produces more outliers than the machine-learning method, while a negative outlier difference means the machine-learning method produces more outliers than ME.

**Table 3**  
Performance Metric Values of MSVR, MLP, and Our CNN Method Based on the Test Set from AR 12665 Collected on 2017 July 13 <sup>a,b</sup>

|                                 |      | $B_{\text{total}}$ | $B_x$              | $B_y$              | $B_z$              | $\phi$             | $\theta$           |
|---------------------------------|------|--------------------|--------------------|--------------------|--------------------|--------------------|--------------------|
| MAE                             | MSVR | 387.23 (9.67)      | 582.00 (65.91)     | 36.15 (9.34)       | 209.48 (21.31)     | 23.09 (1.16)       | 120.30 (23.96)     |
|                                 | MLP  | 108.99 (17.69)     | 90.04 (5.95)       | 76.68 (3.25)       | 66.71 (18.89)      | 7.67 (0.97)        | 23.38 (4.95)       |
|                                 | CNN  | 87.70 (10.69)      | 79.27 (3.60)       | 58.04 (3.05)       | 53.26 (13.44)      | 7.26 (0.80)        | 19.94 (4.25)       |
| PA                              | MSVR | 19.7% (1.4%)       | 5.3% (2.2%)        | 7.8% (1.8%)        | 78.9% (1.3%)       | 9.2% (0.8%)        | 0.5% (0.5%)        |
|                                 | MLP  | 87.0% (2.4%)       | 89.9% (0.9%)       | 94.5% (1.1%)       | 91.9% (1.9%)       | 85.8% (1.8%)       | 51.2% (5.7%)       |
|                                 | CNN  | 90.8% (1.3%)       | 92.4% (0.5%)       | 96.4% (0.8%)       | 93.8% (1.2%)       | 87.7% (1.8%)       | 60.0% (3.7%)       |
| R-squared                       | MSVR | 0.24 (0.27)        | −3.37 (1.53)       | −2.39 (0.56)       | 0.54 (0.12)        | 0.13 (0.09)        | −5.67 (3.98)       |
|                                 | MLP  | 0.85 (0.04)        | 0.77 (0.04)        | 0.71 (0.06)        | 0.86 (0.06)        | 0.68 (0.05)        | 0.49 (0.12)        |
|                                 | CNN  | 0.90 (0.02)        | 0.80 (0.03)        | 0.79 (0.05)        | 0.89 (0.04)        | 0.70 (0.04)        | 0.50 (0.14)        |
| PPMCC <sup>c</sup>              | MSVR | 0.73 (0.10)        | 0.18 (0.06)        | 0.52 (0.08)        | 0.84 (0.03)        | 0.76 (0.04)        | 0.35 (0.14)        |
|                                 | MLP  | 0.95 (0.01)        | 0.89 (0.02)        | 0.86 (0.04)        | 0.94 (0.02)        | 0.84 (0.03)        | 0.71 (0.08)        |
|                                 | CNN  | <b>0.96</b> (0.01) | <b>0.89</b> (0.02) | <b>0.89</b> (0.03) | <b>0.95</b> (0.02) | <b>0.85</b> (0.03) | <b>0.72</b> (0.09) |
| Outlier difference <sup>d</sup> | MSVR | 5448 (1026)        | 6767 (2603)        | 3142 (1633)        | 4052 (864)         | 34672 (7581)       | 93448 (19733)      |
|                                 | MLP  | 5668 (1108)        | 6623 (2620)        | 3127 (1674)        | 4185 (928)         | 34716 (7959)       | 39277 (14562)      |
|                                 | CNN  | 5600 (1128)        | 6267 (2557)        | 2953 (1583)        | 4137 (915)         | 34721 (7945)       | 24276 (12194)      |

**Notes.**<sup>a</sup> Each number in the table represents the average value of 10 experiments.<sup>b</sup> Standard deviations are enclosed in parentheses.<sup>c</sup> The best PPMCC values achieved by the three machine-learning methods are highlighted in bold.<sup>d</sup> A positive outlier difference means ME produces more outliers than the machine-learning method, while a negative outlier difference means the machine-learning method produces more outliers than ME.

or higher for the total magnetic field strength, regardless of whether the training and test data used by the CNN method are from the same or different ARs or are close (e.g., within  $\sim 3$  days) or distant (e.g., over 2 yr) in time. This finding can be seen from Tables A3–A6, where the PPMCC of  $B_{\text{total}}$  in  $D_2^{\text{train}} \rightarrow D_1^{\text{test}}$  ( $D_3^{\text{train}} \rightarrow D_1^{\text{test}}$ ,

$D_1^{\text{train}} \rightarrow D_2^{\text{test}}$ ,  $D_3^{\text{train}} \rightarrow D_2^{\text{test}}$ ,  $D_1^{\text{train}} \rightarrow D_3^{\text{test}}$ ,  $D_2^{\text{train}} \rightarrow D_3^{\text{test}}$ ,  $D_1^{\text{train}} \rightarrow D_4^{\text{test}}$ ,  $D_2^{\text{train}} \rightarrow D_4^{\text{test}}$ , and  $D_3^{\text{train}} \rightarrow D_4^{\text{test}}$ ) is 0.956 (0.924, 0.983, 0.951, 0.976, 0.979, 0.936, 0.927, and 0.896).

2. With respect to the same test image, using the training data from the same AR from which the test image is taken

**Table 4**  
Performance Metric Values of MSVR, MLP, and Our CNN Method Based on the Test Set from AR 12673 Collected on 2017 September 6<sup>a,b</sup>

|                                 |      | $B_{\text{total}}$ | $B_x$              | $B_y$              | $B_z$              | $\phi$             | $\theta$           |
|---------------------------------|------|--------------------|--------------------|--------------------|--------------------|--------------------|--------------------|
| MAE                             | MSVR | 549.84 (0.01)      | 851.67 (0.01)      | 1079.51 (0.01)     | 709.89 (0.01)      | 73.19 (0.01)       | 56.87 (0.01)       |
|                                 | MLP  | 339.40 (8.48)      | 206.18 (6.98)      | 203.56 (5.43)      | 223.20 (5.43)      | 7.35 (0.14)        | 13.23 (0.17)       |
|                                 | CNN  | 198.92 (3.94)      | 150.57 (2.17)      | 128.04 (1.63)      | 139.30 (4.40)      | 5.57 (0.12)        | 9.27 (0.20)        |
| PA                              | MSVR | 17.7% (0.1%)       | 39.7% (0.1%)       | 43.9% (0.1%)       | 6.9% (0.1%)        | 2.3% (0.1%)        | 12.9% (0.1%)       |
|                                 | MLP  | 55.9% (1.3%)       | 70.4% (1.5%)       | 67.9% (0.8%)       | 73.5% (0.8%)       | 82.6% (1.1%)       | 66.0% (0.7%)       |
|                                 | CNN  | 73.6% (0.4%)       | 80.4% (0.4%)       | 84.9% (0.2%)       | 85.7% (1.5%)       | 90.9% (0.3%)       | 78.4% (0.5%)       |
| R-squared                       | MSVR | 0.45 (0.01)        | -1.37 (0.01)       | -7.11 (0.01)       | -0.05 (0.01)       | -5.49 (0.01)       | -1.10 (0.01)       |
|                                 | MLP  | 0.60 (0.01)        | 0.80 (0.01)        | 0.57 (0.01)        | 0.65 (0.01)        | 0.76 (0.01)        | 0.77 (0.01)        |
|                                 | CNN  | 0.84 (0.01)        | 0.87 (0.01)        | 0.79 (0.01)        | 0.74 (0.01)        | 0.78 (0.01)        | 0.80 (0.01)        |
| PPMCC <sup>c</sup>              | MSVR | 0.81 (0.01)        | -0.18 (0.01)       | -0.31 (0.01)       | 0.56 (0.01)        | 0.81 (0.01)        | 0.54 (0.01)        |
|                                 | MLP  | 0.85 (0.01)        | 0.92 (0.01)        | 0.81 (0.01)        | 0.82 (0.01)        | 0.88 (0.01)        | 0.88 (0.01)        |
|                                 | CNN  | <b>0.93</b> (0.01) | <b>0.94</b> (0.01) | <b>0.89</b> (0.01) | <b>0.86</b> (0.01) | <b>0.88</b> (0.01) | <b>0.90</b> (0.01) |
| Outlier difference <sup>d</sup> | MSVR | 19154 (0)          | 21841 (0)          | 15980 (0)          | 12306 (0)          | 21734 (0)          | 32424 (0)          |
|                                 | MLP  | 19632 (20)         | 22346 (20)         | 16780 (38)         | 12941 (8)          | 21918 (10)         | 20692 (552)        |
|                                 | CNN  | 19664 (11)         | 22234 (46)         | 16534 (35)         | 12965 (7)          | 21950 (7)          | 14294 (1436)       |

#### Notes.

<sup>a</sup> Each number in the table represents the average value of 10 experiments.

<sup>b</sup> Standard deviations are enclosed in parentheses.

<sup>c</sup> The best PPMCC values achieved by the three machine-learning methods are highlighted in bold.

<sup>d</sup> A positive outlier difference means ME produces more outliers than the machine-learning method, while a negative outlier difference means the machine-learning method produces more outliers than ME.

yields a better result with a higher PPMCC than using the training and test data that are from different ARs. This finding can be seen from Tables A3 and A4, where the PPMCC of  $B_{\text{total}}$  in  $D_2^{\text{train}} \rightarrow D_1^{\text{test}}$  is 0.956, which is greater than the PPMCC of  $B_{\text{total}}$ , 0.924, in  $D_3^{\text{train}} \rightarrow D_1^{\text{test}}$ . Moreover, the PPMCC of  $B_{\text{total}}$  in  $D_1^{\text{train}} \rightarrow D_2^{\text{test}}$  is 0.983, which is greater than the PPMCC of  $B_{\text{total}}$ , 0.951, in  $D_3^{\text{train}} \rightarrow D_2^{\text{test}}$ .

- However, with respect to the same test image, using the training and test data that are close in time does not necessarily yield a better result than using the training and test data that are distant in time. This finding can be seen from Table A6, where the PPMCC of  $B_{\text{total}}$  in  $D_1^{\text{train}} \rightarrow D_4^{\text{test}}$  is 0.936, which is greater than the PPMCC of  $B_{\text{total}}$ , 0.896, in  $D_3^{\text{train}} \rightarrow D_4^{\text{test}}$ , though  $D_3$  is closer to  $D_4$  than  $D_1$  in time.
- From Tables A3–A6, we can see that the CNN-inferred results have far fewer outliers than the ME-calculated results for all of  $B_{\text{total}}$ ,  $B_x$ ,  $B_y$ ,  $B_z$ ,  $\phi$ , and  $\theta$  in all of the experiments except for  $B_y$  in Table A4. This finding is consistent with the results reported in Table 1.

## 5. Discussion and Conclusions

We develop a new machine-learning method to infer vector magnetic fields from Stokes profiles of GST/NIRIS based on a CNN and the ME method. We then conduct a series of experiments to evaluate the performance of our method. First, we use data samples (pixels) from AR 12371 collected on 2015 June 22 to train the CNN model, where the labels (i.e., vector magnetic fields) of the training data samples are calculated by the ME method. Next, we use the trained model to infer vector magnetic fields from Stokes profiles of pixels in three different unseen test sets. The first test set contains image data from AR 12371 collected on 2015 June 25. The second test set contains

image data from AR 12665 collected on 2017 July 13. The third test set contains image data from AR 12673 collected on 2017 September 6. We compare our CNN method with the ME method and two related machine-learning algorithms, MSVR and MLP, on the three test sets. Finally, we conduct more experiments by varying the training data to get different trained models and applying the models to different test data.

Our findings based on these experiments are consistent and summarized as follows.

- Our CNN method produces smoother and cleaner magnetic maps with fewer outliers (noise pixels) than the ME method.
- It takes  $\sim 50$  s for the CNN method to process an image of  $720 \times 720$  pixels comprising Stokes profiles of GST/NIRIS, which is four to six times faster than the current version of the ME method. The ability to produce vector magnetic fields in nearly real time is essential to space weather forecasting.
- Our CNN-inferred and ME-calculated results are highly correlated and close to each other with a PPMCC of  $\sim 0.9$  or higher for the total magnetic field strength, regardless of whether the training and test data used by the CNN method are from the same or different ARs or are close (e.g., within  $\sim 3$  days) or distant (e.g., over 2 yr) in time. With respect to the same test image, using the training data from the same AR in which the test image is taken yields a better result with a higher PPMCC than using the training and test data that are from different ARs. Hence, for a given test image, it is recommended to adopt the CNN model trained on the same AR from which the test image is collected.
- The CNN-inferred results are closer to the ME-calculated results, with PPMCC values being closer to 1, on average, than those from the related machine-learning methods MSVR and MLP. In particular, the CNN method

outperforms the current best machine-learning method (MLP) by 2.6%, on average, in PPMCC. This happens because the CNN method is able to exploit the spatial information of the Stokes profiles and learn latent patterns between the Stokes profiles and ME-calculated vector magnetic fields in a better way.

Based on these findings, we conclude that the proposed CNN model can be considered as an alternative, efficient method for Stokes inversion for high-resolution polarimetric observations obtained by GST/NIRIS. More accurate and efficient Stokes inversion will improve nearly real-time prediction of space weather in the future as it prepares more accurate magnetic boundary conditions at the solar surface quickly. With the advent of big and complex observational data gathered from diverse instruments, such as the BBSO/GST and the upcoming Daniel K. Inouye Solar Telescope, it is expected that our physics-assisted deep learning-based CNN tool will be a useful utility for processing and analyzing the data.

We thank the referees for very helpful and thoughtful comments. The data used in this study were obtained with the GST at BBSO, which is operated by the New Jersey Institute of Technology. Obtaining the excellent data would not have been possible without the help of the BBSO team. The BBSO operation is supported by NJIT and NSF grant AGS-1821294. The GST operation is partly supported by the Korea

Astronomy and Space Science Institute and Seoul National University. The related machine-learning algorithms studied here were implemented in Python. This work was supported by NSF grant AGS-1927578. Y.X., J.J., C.L., and H.W. acknowledge the support of NASA under grants NNX16AF72G, 80NSSC17K0016, 80NSSC18K0673, and 80NSSC18K1705.

*Facility:* Big Bear Solar Observatory.

## Appendix

Tables A1 and A2 present the performance metric values of our CNN method based on the test images from AR 12371 and AR 12665 collected at 10 different time points on 2015 June 25 and 2017 July 13, respectively, where training data were taken from AR 12371 on 2015 June 22. Tables A3–A6 present the performance metric values of our CNN method obtained by using different combinations of ARs as training data and using the images from AR 12371 on 2015 June 22 17:33:00 UT, AR 12371 on 2015 June 25 20:00:00 UT, AR 12665 on 2017 July 13 18:35:00 UT, and AR 12673 on 2017 September 6 19:18:00 UT, respectively, as test data. Figure A1 (Figures A2 and A3) presents results for the quantities  $B_x$ ,  $B_y$ , and  $B_z$ , displayed from top to bottom in the figure, based on the test image from AR 12371 (AR 12665 and AR 12673) collected on 2015 June 25 20:00:00 UT (2017 July 13 18:35:00 UT and 2017 September 6 19:18:00 UT) where training data were taken from AR 12371 on 2015 June 22.

**Table A1**  
Performance Metric Values of Our CNN Method Based on the Test Images from AR 12371 Collected at 10 Different Time Points on 2015 June 25

|                        |             |                    | $B_{\text{total}}$ | $B_x$  | $B_y$  | $B_z$  | $\phi$ | $\theta$ |
|------------------------|-------------|--------------------|--------------------|--------|--------|--------|--------|----------|
| 2015 Jun 25 (AR 12371) | 17:02:00 UT | MAE                | 79.916             | 72.753 | 61.458 | 53.999 | 4.578  | 9.069    |
|                        |             | PA                 | 91.9%              | 93.4%  | 95.3%  | 94.6%  | 93.6%  | 81.3%    |
|                        |             | $R$ -squared       | 0.965              | 0.935  | 0.921  | 0.968  | 0.823  | 0.769    |
|                        |             | PPMCC              | 0.983              | 0.968  | 0.961  | 0.984  | 0.908  | 0.879    |
|                        |             | Outlier difference | 3935               | 3881   | 159    | 1857   | 14543  | 18746    |
|                        | 17:20:00 UT | MAE                | 86.211             | 74.157 | 56.345 | 54.512 | 4.254  | 8.416    |
|                        |             | PA                 | 90.2%              | 92.0%  | 96.6%  | 94.4%  | 94.1%  | 81.9%    |
|                        |             | $R$ -squared       | 0.961              | 0.941  | 0.932  | 0.966  | 0.833  | 0.8797   |
|                        |             | PPMCC              | 0.982              | 0.971  | 0.968  | 0.984  | 0.913  | 0.895    |
|                        |             | Outlier difference | 5148               | 3754   | −263   | 2225   | 13631  | 14329    |
|                        | 17:41:00 UT | MAE                | 75.772             | 68.146 | 57.255 | 47.984 | 4.184  | 8.376    |
|                        |             | PA                 | 92.4%              | 93.6%  | 96.1%  | 95.5%  | 93.8%  | 82.5%    |
|                        |             | $R$ -squared       | 0.968              | 0.947  | 0.925  | 0.974  | 0.847  | 0.795    |
|                        |             | PPMCC              | 0.985              | 0.974  | 0.963  | 0.987  | 0.920  | 0.893    |
|                        |             | Outlier difference | 2958               | 2995   | −555   | 1596   | 12989  | 13302    |
|                        | 18:00:00 UT | MAE                | 77.941             | 71.767 | 57.802 | 50.538 | 4.498  | 8.586    |
|                        |             | PA                 | 92.6%              | 93.6%  | 96.7%  | 95.4%  | 93.9%  | 82.4%    |
|                        |             | $R$ -squared       | 0.970              | 0.943  | 0.927  | 0.974  | 0.825  | 0.786    |
|                        |             | PPMCC              | 0.987              | 0.972  | 0.966  | 0.987  | 0.909  | 0.888    |
|                        |             | Outlier difference | 2813               | 3036   | −630   | 1639   | 14735  | 15265    |
|                        | 18:20:00 UT | MAE                | 80.294             | 74.324 | 56.363 | 49.600 | 4.278  | 8.389    |
|                        |             | PA                 | 91.9%              | 92.5%  | 96.5%  | 95.7%  | 94.5%  | 83.2%    |
|                        |             | $R$ -squared       | 0.965              | 0.940  | 0.931  | 0.975  | 0.839  | 0.788    |
|                        |             | PPMCC              | 0.984              | 0.970  | 0.967  | 0.988  | 0.917  | 0.889    |
|                        |             | Outlier difference | 2754               | 2840   | −432   | 1407   | 13432  | 12845    |
|                        | 18:40:00 UT | MAE                | 82.176             | 77.420 | 57.013 | 52.885 | 4.760  | 8.948    |
|                        |             | PA                 | 91.2%              | 92.0%  | 96.4%  | 94.8%  | 93.7%  | 82.2%    |
|                        |             | $R$ -squared       | 0.964              | 0.942  | 0.933  | 0.972  | 0.822  | 0.770    |
|                        |             | PPMCC              | 0.984              | 0.971  | 0.967  | 0.987  | 0.907  | 0.881    |
|                        |             | Outlier difference | 2508               | 2863   | −389   | 1253   | 15026  | 10675    |

**Table A1**  
(Continued)

|             |                    |  | $B_{\text{total}}$ | $B_x$  | $B_y$  | $B_z$  | $\phi$ | $\theta$ |
|-------------|--------------------|--|--------------------|--------|--------|--------|--------|----------|
| 19:00:00 UT | MAE                |  | 79.144             | 76.014 | 57.960 | 52.950 | 4.716  | 9.507    |
|             | PA                 |  | 91.7%              | 92.8%  | 96.1%  | 94.9%  | 93.7%  | 80.8%    |
|             | $R$ -squared       |  | 0.967              | 0.943  | 0.925  | 0.975  | 0.828  | 0.757    |
|             | PPMCC              |  | 0.985              | 0.972  | 0.964  | 0.990  | 0.911  | 0.872    |
|             | Outlier difference |  | 2450               | 3166   | −424   | 1125   | 15383  | 11646    |
| 19:22:00 UT | MAE                |  | 86.917             | 79.777 | 59.833 | 51.695 | 4.470  | 9.586    |
|             | PA                 |  | 90.6%              | 92.3%  | 95.9%  | 95.4%  | 93.3%  | 81.9%    |
|             | $R$ -squared       |  | 0.962              | 0.939  | 0.924  | 0.975  | 0.837  | 0.742    |
|             | PPMCC              |  | 0.983              | 0.970  | 0.964  | 0.988  | 0.915  | 0.865    |
|             | Outlier difference |  | 2644               | 3508   | −392   | 1297   | 14243  | 11326    |
| 19:41:00 UT | MAE                |  | 80.683             | 82.218 | 58.095 | 51.991 | 4.775  | 11.341   |
|             | PA                 |  | 91.7%              | 91.8%  | 96.3%  | 95.5%  | 93.1%  | 78.8%    |
|             | $R$ -squared       |  | 0.966              | 0.935  | 0.928  | 0.974  | 0.827  | 0.706    |
|             | PPMCC              |  | 0.984              | 0.968  | 0.965  | 0.988  | 0.910  | 0.845    |
|             | Outlier difference |  | 2426               | 3722   | −495   | 908    | 15939  | 11092    |
| 20:00:00 UT | MAE                |  | 86.660             | 88.997 | 66.140 | 55.653 | 4.867  | 11.136   |
|             | PA                 |  | 91.6%              | 91.3%  | 95.2%  | 94.7%  | 92.2%  | 79.1%    |
|             | $R$ -squared       |  | 0.963              | 0.936  | 0.901  | 0.976  | 0.838  | 0.720    |
|             | PPMCC              |  | 0.983              | 0.968  | 0.951  | 0.989  | 0.916  | 0.853    |
|             | Outlier difference |  | 2959               | 4380   | −770   | 1050   | 15108  | 7219     |

**Table A2**

Performance Metric Values of Our CNN Method Based on the Test Images from AR 12665 Collected at 10 Different Time Points on 2017 July 13

|                        |             |                    | $B_{\text{total}}$ | $B_x$  | $B_y$  | $B_z$  | $\phi$ | $\theta$ |
|------------------------|-------------|--------------------|--------------------|--------|--------|--------|--------|----------|
| 2017 Jul 13 (AR 12665) | 17:18:00 UT | MAE                | 96.763             | 77.228 | 59.555 | 65.277 | 6.995  | 15.623   |
|                        |             | PA                 | 89.9%              | 93.5%  | 95.7%  | 92.7%  | 88.5%  | 62.2%    |
|                        |             | $R$ -squared       | 0.895              | 0.796  | 0.726  | 0.875  | 0.716  | 0.686    |
|                        |             | PPMCC              | 0.961              | 0.893  | 0.857  | 0.947  | 0.851  | 0.834    |
|                        |             | Outlier difference | 5612               | 8931   | 5341   | 4805   | 36440  | 39816    |
|                        | 17:54:00 UT | MAE                | 108.101            | 86.635 | 60.292 | 83.230 | 8.276  | 16.899   |
|                        |             | PA                 | 90.8%              | 92.3%  | 95.6%  | 92.8%  | 86.0%  | 60.8%    |
|                        |             | $R$ -squared       | 0.866              | 0.745  | 0.695  | 0.789  | 0.647  | 0.677    |
|                        |             | PPMCC              | 0.953              | 0.864  | 0.838  | 0.902  | 0.814  | 0.829    |
|                        |             | Outlier difference | 5430               | 11516  | 5702   | 5728   | 45541  | 41897    |
|                        | 18:25:00 UT | MAE                | 95.509             | 81.222 | 59.119 | 66.639 | 7.984  | 17.809   |
|                        |             | PA                 | 89.6%              | 92.2%  | 95.7%  | 91.9%  | 86.5%  | 60.1%    |
|                        |             | $R$ -squared       | 0.914              | 0.822  | 0.792  | 0.874  | 0.661  | 0.664    |
|                        |             | PPMCC              | 0.971              | 0.907  | 0.893  | 0.947  | 0.824  | 0.820    |
|                        |             | Outlier difference | 3874               | 8158   | 4134   | 3657   | 42169  | 39937    |
|                        | 18:35:00 UT | MAE                | 73.684             | 71.555 | 51.170 | 49.023 | 7.573  | 17.437   |
|                        |             | PA                 | 91.5%              | 93.3%  | 96.4%  | 92.6%  | 84.8%  | 60.6%    |
|                        |             | $R$ -squared       | 0.950              | 0.841  | 0.851  | 0.941  | 0.663  | 0.665    |
|                        |             | PPMCC              | 0.976              | 0.918  | 0.926  | 0.971  | 0.827  | 0.821    |
|                        |             | Outlier difference | 3801               | 7280   | 3413   | 2478   | 35649  | 28274    |
|                        | 20:19:00 UT | MAE                | 75.811             | 78.550 | 55.695 | 38.701 | 8.014  | 26.263   |
|                        |             | PA                 | 92.8%              | 92.1%  | 97.4%  | 95.7%  | 86.6%  | 55.6%    |
|                        |             | $R$ -squared       | 0.915              | 0.826  | 0.831  | 0.930  | 0.680  | 0.456    |
|                        |             | PPMCC              | 0.960              | 0.910  | 0.916  | 0.966  | 0.831  | 0.693    |
|                        |             | Outlier difference | 5089               | 4479   | 1900   | 3341   | 41315  | 12610    |
|                        | 20:52:00 UT | MAE                | 77.201             | 78.757 | 56.624 | 41.618 | 7.979  | 26.759   |
|                        |             | PA                 | 92.6%              | 92.1%  | 97.6%  | 95.1%  | 86.4%  | 54.3%    |
|                        |             | $R$ -squared       | 0.914              | 0.805  | 0.827  | 0.926  | 0.682  | 0.401    |
|                        |             | PPMCC              | 0.957              | 0.897  | 0.913  | 0.963  | 0.834  | 0.656    |
|                        |             | Outlier difference | 5878               | 4499   | 1788   | 4019   | 38418  | 10550    |
|                        | 21:20:00 UT | MAE                | 80.011             | 77.987 | 57.099 | 42.847 | 7.200  | 25.012   |
|                        |             | PA                 | 92.2%              | 92.4%  | 97.4%  | 94.9%  | 88.3%  | 55.1%    |

**Table A2**  
(Continued)

|             |                    | $B_{\text{total}}$ | $B_x$  | $B_y$  | $B_z$  | $\phi$ | $\theta$ |
|-------------|--------------------|--------------------|--------|--------|--------|--------|----------|
|             | $R$ -squared       | 0.901              | 0.799  | 0.805  | 0.918  | 0.710  | 0.352    |
|             | PPMCC              | 0.952              | 0.895  | 0.901  | 0.961  | 0.851  | 0.624    |
|             | Outlier difference | 5997               | 4505   | 1771   | 3992   | 33783  | 15985    |
| 21:48:00 UT | MAE                | 84.746             | 78.946 | 57.296 | 43.822 | 6.471  | 21.160   |
|             | PA                 | 89.6%              | 91.9%  | 97.0%  | 94.9%  | 90.1%  | 60.7%    |
|             | $R$ -squared       | 0.895              | 0.791  | 0.808  | 0.917  | 0.728  | 0.380    |
|             | PPMCC              | 0.953              | 0.891  | 0.901  | 0.963  | 0.859  | 0.643    |
|             | Outlier difference | 6234               | 4556   | 1938   | 3916   | 28967  | 19769    |
|             |                    |                    |        |        |        |        |          |
| 22:18:00 UT | MAE                | 95.769             | 81.151 | 61.962 | 52.672 | 5.890  | 16.784   |
|             | PA                 | 88.8%              | 91.7%  | 95.7%  | 93.6%  | 90.3%  | 64.6%    |
|             | $R$ -squared       | 0.869              | 0.771  | 0.779  | 0.893  | 0.776  | 0.364    |
|             | PPMCC              | 0.942              | 0.881  | 0.884  | 0.952  | 0.888  | 0.640    |
|             | Outlier difference | 7325               | 4238   | 1721   | 5022   | 22740  | 15454    |
| 22:39:00 UT | MAE                | 89.352             | 80.647 | 61.617 | 48.760 | 6.226  | 15.683   |
|             | PA                 | 90.3%              | 92.0%  | 95.8%  | 94.4%  | 89.4%  | 65.9%    |
|             | $R$ -squared       | 0.889              | 0.774  | 0.751  | 0.913  | 0.775  | 0.399    |
|             | PPMCC              | 0.951              | 0.885  | 0.868  | 0.961  | 0.889  | 0.664    |
|             | Outlier difference | 6757               | 4506   | 1826   | 4408   | 22186  | 18471    |

**Table A3**Performance Metric Values of Our CNN Method Obtained by Using  $D_1$  to Form Test Data and Different Combinations of  $D_2$ ,  $D_3$ , and  $D_4$  to Form Training Data

|                    |                                                          | $B_{\text{total}}$ | $B_x$  | $B_y$   | $B_z$   | $\phi$ | $\theta$ |
|--------------------|----------------------------------------------------------|--------------------|--------|---------|---------|--------|----------|
| MAE                | $D_2^{\text{train}} \rightarrow D_1^{\text{test}}$       | 112.104            | 70.871 | 77.554  | 83.761  | 5.040  | 10.286   |
|                    | $D_3^{\text{train}} \rightarrow D_1^{\text{test}}$       | 168.905            | 96.727 | 116.505 | 112.322 | 5.724  | 11.874   |
|                    | $D_{2,3}^{\text{train}} \rightarrow D_1^{\text{test}}$   | 99.187             | 74.859 | 75.777  | 81.588  | 5.330  | 10.330   |
|                    | $D_{2,4}^{\text{train}} \rightarrow D_1^{\text{test}}$   | 96.981             | 78.739 | 77.672  | 70.458  | 5.111  | 11.356   |
|                    | $D_{3,4}^{\text{train}} \rightarrow D_1^{\text{test}}$   | 137.511            | 87.619 | 105.794 | 83.560  | 5.315  | 11.086   |
|                    | $D_{2,3,4}^{\text{train}} \rightarrow D_1^{\text{test}}$ | 97.594             | 73.092 | 75.116  | 74.675  | 5.095  | 10.258   |
| PA                 | $D_2^{\text{train}} \rightarrow D_1^{\text{test}}$       | 88.5%              | 93.6%  | 92.4%   | 90.8%   | 90.6%  | 78.0%    |
|                    | $D_3^{\text{train}} \rightarrow D_1^{\text{test}}$       | 71.8%              | 89.5%  | 80.8%   | 85.7%   | 89.0%  | 78.6%    |
|                    | $D_{2,3}^{\text{train}} \rightarrow D_1^{\text{test}}$   | 89.2%              | 92.7%  | 92.5%   | 91.2%   | 89.6%  | 78.5%    |
|                    | $D_{2,4}^{\text{train}} \rightarrow D_1^{\text{test}}$   | 90.0%              | 92.2%  | 92.6%   | 92.4%   | 90.5%  | 76.3%    |
|                    | $D_{3,4}^{\text{train}} \rightarrow D_1^{\text{test}}$   | 81.7%              | 91.3%  | 87.6%   | 90.5%   | 90.2%  | 78.9%    |
|                    | $D_{2,3,4}^{\text{train}} \rightarrow D_1^{\text{test}}$ | 89.7%              | 92.7%  | 92.5%   | 92.3%   | 90.2%  | 79.1%    |
| $R$ -squared       | $D_2^{\text{train}} \rightarrow D_1^{\text{test}}$       | 0.903              | 0.913  | 0.878   | 0.955   | 0.867  | 0.710    |
|                    | $D_3^{\text{train}} \rightarrow D_1^{\text{test}}$       | 0.845              | 0.860  | 0.810   | 0.929   | 0.867  | 0.576    |
|                    | $D_{2,3}^{\text{train}} \rightarrow D_1^{\text{test}}$   | 0.907              | 0.910  | 0.875   | 0.953   | 0.868  | 0.706    |
|                    | $D_{2,4}^{\text{train}} \rightarrow D_1^{\text{test}}$   | 0.909              | 0.899  | 0.862   | 0.962   | 0.861  | 0.657    |
|                    | $D_{3,4}^{\text{train}} \rightarrow D_1^{\text{test}}$   | 0.886              | 0.888  | 0.830   | 0.954   | 0.864  | 0.661    |
|                    | $D_{2,3,4}^{\text{train}} \rightarrow D_1^{\text{test}}$ | 0.904              | 0.908  | 0.874   | 0.956   | 0.869  | 0.701    |
| PPMCC              | $D_2^{\text{train}} \rightarrow D_1^{\text{test}}$       | 0.956              | 0.956  | 0.937   | 0.982   | 0.935  | 0.847    |
|                    | $D_3^{\text{train}} \rightarrow D_1^{\text{test}}$       | 0.924              | 0.933  | 0.929   | 0.965   | 0.933  | 0.780    |
|                    | $D_{2,3}^{\text{train}} \rightarrow D_1^{\text{test}}$   | 0.954              | 0.956  | 0.936   | 0.980   | 0.936  | 0.846    |
|                    | $D_{2,4}^{\text{train}} \rightarrow D_1^{\text{test}}$   | 0.954              | 0.951  | 0.932   | 0.982   | 0.932  | 0.822    |
|                    | $D_{3,4}^{\text{train}} \rightarrow D_1^{\text{test}}$   | 0.946              | 0.947  | 0.931   | 0.978   | 0.932  | 0.821    |
|                    | $D_{2,3,4}^{\text{train}} \rightarrow D_1^{\text{test}}$ | 0.952              | 0.955  | 0.935   | 0.980   | 0.936  | 0.843    |
| Outlier difference | $D_2^{\text{train}} \rightarrow D_1^{\text{test}}$       | 9396               | 4718   | 3419    | 6808    | 33687  | 22528    |
|                    | $D_3^{\text{train}} \rightarrow D_1^{\text{test}}$       | 9527               | 3625   | 3355    | 6554    | 33202  | 27896    |
|                    | $D_{2,3}^{\text{train}} \rightarrow D_1^{\text{test}}$   | 9266               | 4045   | 3120    | 6760    | 33114  | 25720    |
|                    | $D_{2,4}^{\text{train}} \rightarrow D_1^{\text{test}}$   | 9185               | 4480   | 3086    | 6782    | 33754  | 24704    |
|                    | $D_{3,4}^{\text{train}} \rightarrow D_1^{\text{test}}$   | 9160               | 3921   | 2878    | 6771    | 33671  | 33627    |
|                    | $D_{2,3,4}^{\text{train}} \rightarrow D_1^{\text{test}}$ | 8668               | 4528   | 3301    | 6813    | 33691  | 26670    |

**Table A4**Performance Metric Values of Our CNN Method Obtained by Using  $D_2$  to Form Test Data and Different Combinations of  $D_1$ ,  $D_3$ , and  $D_4$  to Form Training Data

|                    |                                                          | $B_{\text{total}}$ | $B_x$   | $B_y$  | $B_z$   | $\phi$ | $\theta$ |
|--------------------|----------------------------------------------------------|--------------------|---------|--------|---------|--------|----------|
| MAE                | $D_1^{\text{train}} \rightarrow D_2^{\text{test}}$       | 86.660             | 88.997  | 66.140 | 55.653  | 4.867  | 11.136   |
|                    | $D_3^{\text{train}} \rightarrow D_2^{\text{test}}$       | 165.558            | 132.399 | 92.024 | 109.629 | 6.157  | 12.191   |
|                    | $D_{1,3}^{\text{train}} \rightarrow D_2^{\text{test}}$   | 83.631             | 86.654  | 61.893 | 51.414  | 4.650  | 10.949   |
|                    | $D_{1,4}^{\text{train}} \rightarrow D_2^{\text{test}}$   | 90.098             | 88.494  | 63.458 | 60.282  | 4.935  | 10.595   |
|                    | $D_{3,4}^{\text{train}} \rightarrow D_2^{\text{test}}$   | 133.132            | 104.756 | 85.091 | 79.925  | 5.250  | 11.805   |
|                    | $D_{1,3,4}^{\text{train}} \rightarrow D_2^{\text{test}}$ | 79.830             | 84.662  | 59.412 | 50.448  | 4.736  | 11.023   |
| PA                 | $D_1^{\text{train}} \rightarrow D_2^{\text{test}}$       | 91.6%              | 91.3%   | 95.2%  | 94.7%   | 92.2%  | 79.1%    |
|                    | $D_3^{\text{train}} \rightarrow D_2^{\text{test}}$       | 69.4%              | 81.8%   | 87.8%  | 81.2%   | 87.3%  | 74.9%    |
|                    | $D_{1,3}^{\text{train}} \rightarrow D_2^{\text{test}}$   | 89.7%              | 90.5%   | 95.5%  | 95.3%   | 93.3%  | 79.2%    |
|                    | $D_{1,4}^{\text{train}} \rightarrow D_2^{\text{test}}$   | 89.0%              | 90.5%   | 95.1%  | 94.0%   | 92.6%  | 80.2%    |
|                    | $D_{3,4}^{\text{train}} \rightarrow D_2^{\text{test}}$   | 79.0%              | 88.1%   | 89.7%  | 89.7%   | 92.3%  | 76.8%    |
|                    | $D_{1,3,4}^{\text{train}} \rightarrow D_2^{\text{test}}$ | 91.9%              | 92.0%   | 95.7%  | 96.0%   | 93.1%  | 78.8%    |
| $R$ -squared       | $D_1^{\text{train}} \rightarrow D_2^{\text{test}}$       | 0.963              | 0.936   | 0.901  | 0.976   | 0.838  | 0.720    |
|                    | $D_3^{\text{train}} \rightarrow D_2^{\text{test}}$       | 0.893              | 0.899   | 0.850  | 0.928   | 0.828  | 0.695    |
|                    | $D_{1,3}^{\text{train}} \rightarrow D_2^{\text{test}}$   | 0.962              | 0.937   | 0.914  | 0.979   | 0.844  | 0.724    |
|                    | $D_{1,4}^{\text{train}} \rightarrow D_2^{\text{test}}$   | 0.956              | 0.936   | 0.907  | 0.972   | 0.839  | 0.727    |
|                    | $D_{3,4}^{\text{train}} \rightarrow D_2^{\text{test}}$   | 0.937              | 0.927   | 0.858  | 0.964   | 0.837  | 0.711    |
|                    | $D_{1,3,4}^{\text{train}} \rightarrow D_2^{\text{test}}$ | 0.966              | 0.938   | 0.918  | 0.980   | 0.842  | 0.724    |
| PPMCC              | $D_1^{\text{train}} \rightarrow D_2^{\text{test}}$       | 0.983              | 0.968   | 0.951  | 0.989   | 0.916  | 0.853    |
|                    | $D_3^{\text{train}} \rightarrow D_2^{\text{test}}$       | 0.951              | 0.949   | 0.939  | 0.982   | 0.915  | 0.846    |
|                    | $D_{1,3}^{\text{train}} \rightarrow D_2^{\text{test}}$   | 0.982              | 0.968   | 0.957  | 0.990   | 0.919  | 0.855    |
|                    | $D_{1,4}^{\text{train}} \rightarrow D_2^{\text{test}}$   | 0.981              | 0.968   | 0.955  | 0.988   | 0.916  | 0.856    |
|                    | $D_{3,4}^{\text{train}} \rightarrow D_2^{\text{test}}$   | 0.982              | 0.968   | 0.954  | 0.989   | 0.916  | 0.850    |
|                    | $D_{1,3,4}^{\text{train}} \rightarrow D_2^{\text{test}}$ | 0.984              | 0.969   | 0.960  | 0.990   | 0.918  | 0.855    |
| Outlier difference | $D_1^{\text{train}} \rightarrow D_2^{\text{test}}$       | 2959               | 4380    | -770   | 1050    | 15108  | 7219     |
|                    | $D_3^{\text{train}} \rightarrow D_2^{\text{test}}$       | 2950               | 3904    | -246   | 1032    | 15054  | 10038    |
|                    | $D_{1,3}^{\text{train}} \rightarrow D_2^{\text{test}}$   | 2948               | 4354    | -666   | 1053    | 15108  | 7392     |
|                    | $D_{1,4}^{\text{train}} \rightarrow D_2^{\text{test}}$   | 2954               | 4231    | -574   | 1055    | 15108  | 11235    |
|                    | $D_{3,4}^{\text{train}} \rightarrow D_2^{\text{test}}$   | 2953               | 3926    | -533   | 1057    | 15000  | 9161     |
|                    | $D_{1,3,4}^{\text{train}} \rightarrow D_2^{\text{test}}$ | 2959               | 4380    | -631   | 1053    | 15108  | 9410     |

**Table A5**Performance Metric Values of Our CNN Method Obtained by Using  $D_3$  to Form Test Data and Different Combinations of  $D_1$ ,  $D_2$ , and  $D_4$  to Form Training Data

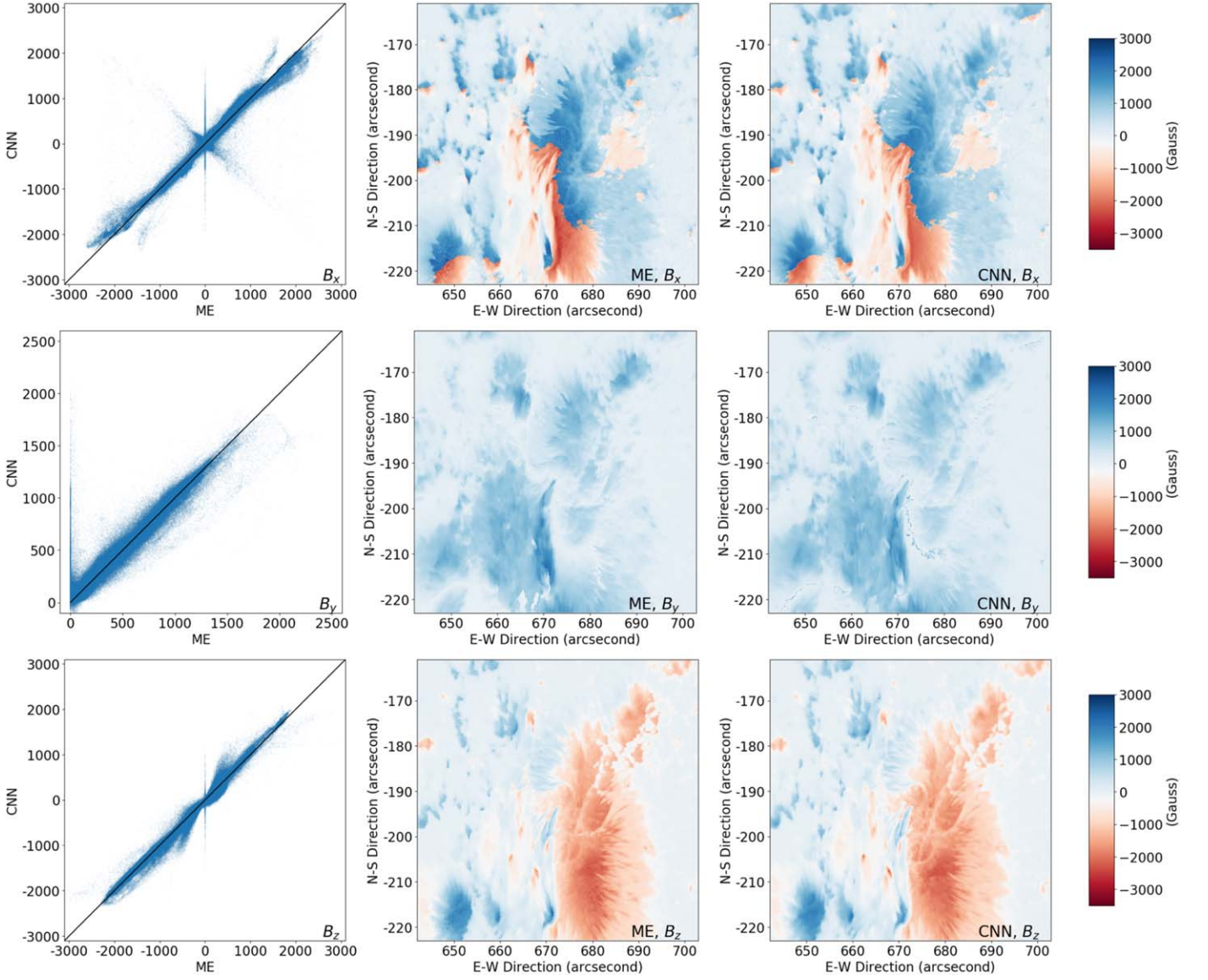
|              |                                                          | $B_{\text{total}}$ | $B_x$  | $B_y$  | $B_z$  | $\phi$ | $\theta$ |
|--------------|----------------------------------------------------------|--------------------|--------|--------|--------|--------|----------|
| MAE          | $D_1^{\text{train}} \rightarrow D_3^{\text{test}}$       | 73.683             | 71.555 | 51.170 | 49.023 | 7.573  | 17.437   |
|              | $D_2^{\text{train}} \rightarrow D_3^{\text{test}}$       | 98.412             | 83.441 | 55.674 | 58.326 | 7.330  | 18.232   |
|              | $D_{1,2}^{\text{train}} \rightarrow D_3^{\text{test}}$   | 70.574             | 68.776 | 48.467 | 43.919 | 7.381  | 16.780   |
|              | $D_{1,4}^{\text{train}} \rightarrow D_3^{\text{test}}$   | 68.492             | 66.340 | 48.593 | 48.398 | 7.394  | 15.661   |
|              | $D_{2,4}^{\text{train}} \rightarrow D_3^{\text{test}}$   | 68.903             | 64.068 | 44.475 | 46.860 | 6.539  | 14.911   |
|              | $D_{1,2,4}^{\text{train}} \rightarrow D_3^{\text{test}}$ | 68.876             | 67.419 | 48.227 | 43.239 | 7.250  | 16.839   |
| PA           | $D_1^{\text{train}} \rightarrow D_3^{\text{test}}$       | 91.5%              | 93.3%  | 96.4%  | 92.6%  | 84.8%  | 60.6%    |
|              | $D_2^{\text{train}} \rightarrow D_3^{\text{test}}$       | 90.1%              | 92.3%  | 95.7%  | 93.9%  | 87.0%  | 54.8%    |
|              | $D_{1,2}^{\text{train}} \rightarrow D_3^{\text{test}}$   | 92.8%              | 93.8%  | 96.6%  | 94.1%  | 85.6%  | 61.3%    |
|              | $D_{1,4}^{\text{train}} \rightarrow D_3^{\text{test}}$   | 91.1%              | 93.1%  | 95.8%  | 93.5%  | 84.2%  | 65.7%    |
|              | $D_{2,4}^{\text{train}} \rightarrow D_3^{\text{test}}$   | 93.1%              | 93.7%  | 96.7%  | 93.8%  | 87.1%  | 68.0%    |
|              | $D_{1,2,4}^{\text{train}} \rightarrow D_3^{\text{test}}$ | 91.5%              | 93.9%  | 96.7%  | 93.1%  | 86.2%  | 61.9%    |
| $R$ -squared | $D_1^{\text{train}} \rightarrow D_3^{\text{test}}$       | 0.950              | 0.841  | 0.851  | 0.941  | 0.663  | 0.665    |
|              | $D_2^{\text{train}} \rightarrow D_3^{\text{test}}$       | 0.926              | 0.830  | 0.850  | 0.924  | 0.661  | 0.678    |
|              | $D_{1,2}^{\text{train}} \rightarrow D_3^{\text{test}}$   | 0.957              | 0.849  | 0.857  | 0.955  | 0.665  | 0.688    |
|              | $D_{1,4}^{\text{train}} \rightarrow D_3^{\text{test}}$   | 0.951              | 0.848  | 0.855  | 0.944  | 0.667  | 0.698    |

**Table A5**  
(Continued)

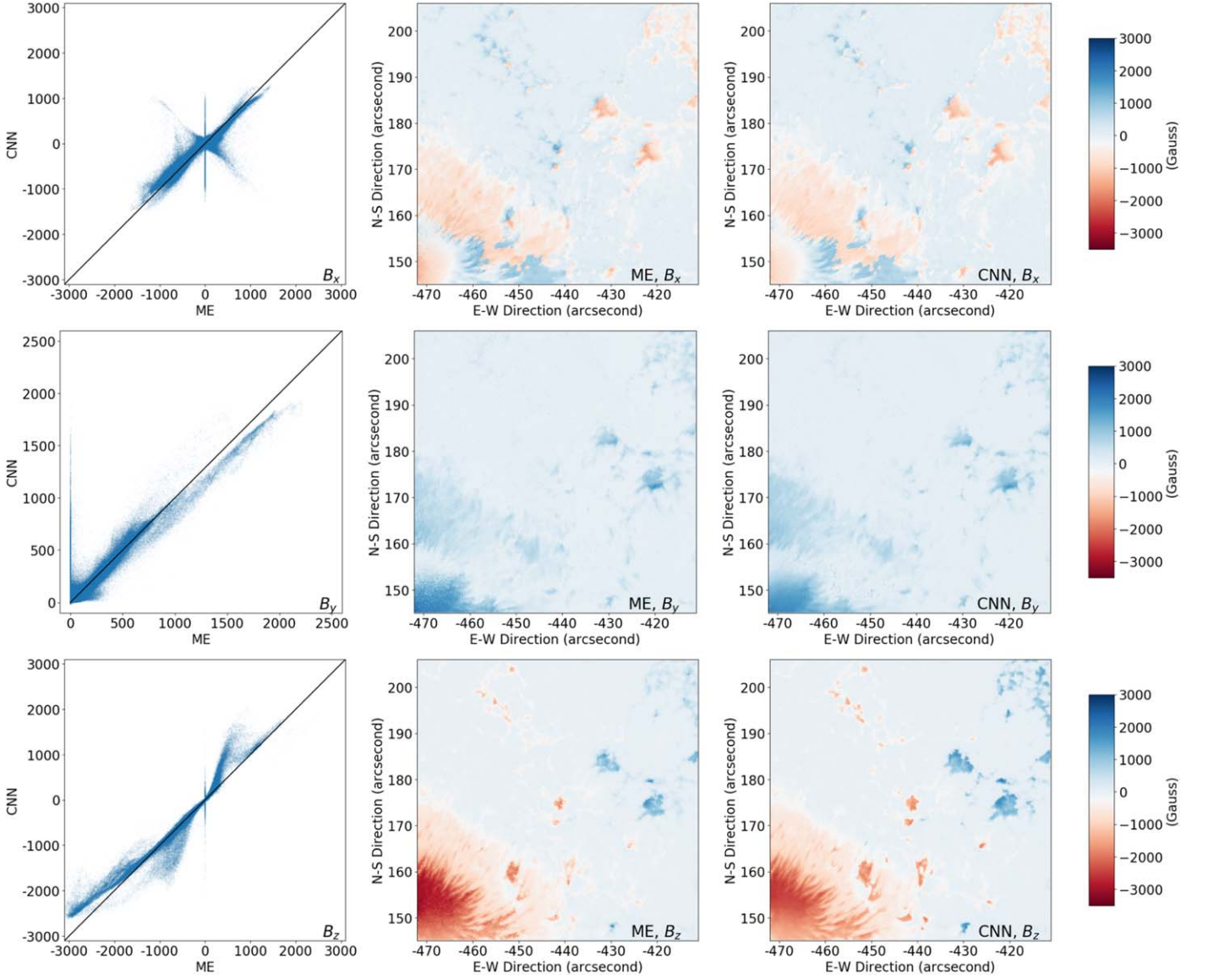
|                    |                                                          | $B_{\text{total}}$ | $B_x$ | $B_y$ | $B_z$ | $\phi$ | $\theta$ |
|--------------------|----------------------------------------------------------|--------------------|-------|-------|-------|--------|----------|
|                    | $D_{2,4}^{\text{train}} \rightarrow D_3^{\text{test}}$   | 0.952              | 0.845 | 0.860 | 0.946 | 0.703  | 0.696    |
|                    | $D_{1,2,4}^{\text{train}} \rightarrow D_3^{\text{test}}$ | 0.959              | 0.848 | 0.858 | 0.956 | 0.672  | 0.680    |
| PPMCC              | $D_1^{\text{train}} \rightarrow D_3^{\text{test}}$       | 0.976              | 0.918 | 0.926 | 0.971 | 0.827  | 0.821    |
|                    | $D_2^{\text{train}} \rightarrow D_3^{\text{test}}$       | 0.979              | 0.913 | 0.923 | 0.978 | 0.829  | 0.829    |
|                    | $D_{1,2}^{\text{train}} \rightarrow D_3^{\text{test}}$   | 0.979              | 0.921 | 0.928 | 0.978 | 0.828  | 0.834    |
|                    | $D_{1,4}^{\text{train}} \rightarrow D_3^{\text{test}}$   | 0.975              | 0.921 | 0.926 | 0.973 | 0.828  | 0.839    |
|                    | $D_{2,4}^{\text{train}} \rightarrow D_3^{\text{test}}$   | 0.976              | 0.922 | 0.929 | 0.973 | 0.843  | 0.844    |
|                    | $D_{1,2,4}^{\text{train}} \rightarrow D_3^{\text{test}}$ | 0.980              | 0.921 | 0.928 | 0.978 | 0.831  | 0.830    |
| Outlier difference | $D_1^{\text{train}} \rightarrow D_3^{\text{test}}$       | 3801               | 7280  | 3413  | 2478  | 35649  | 28274    |
|                    | $D_2^{\text{train}} \rightarrow D_3^{\text{test}}$       | 3837               | 7284  | 3545  | 2497  | 35661  | 27447    |
|                    | $D_{1,2}^{\text{train}} \rightarrow D_3^{\text{test}}$   | 3800               | 7252  | 3433  | 2480  | 35647  | 25888    |
|                    | $D_{1,4}^{\text{train}} \rightarrow D_3^{\text{test}}$   | 3812               | 7200  | 3405  | 2481  | 35645  | 31320    |
|                    | $D_{2,4}^{\text{train}} \rightarrow D_3^{\text{test}}$   | 3824               | 6873  | 3494  | 2496  | 35657  | 26492    |
|                    | $D_{1,2,4}^{\text{train}} \rightarrow D_3^{\text{test}}$ | 3801               | 7371  | 3463  | 2490  | 35645  | 24124    |

**Table A6**Performance Metric Values of Our CNN Method Obtained by Using  $D_4$  to Form Test Data and Different Combinations of  $D_1$ ,  $D_2$ , and  $D_3$  to Form Training Data

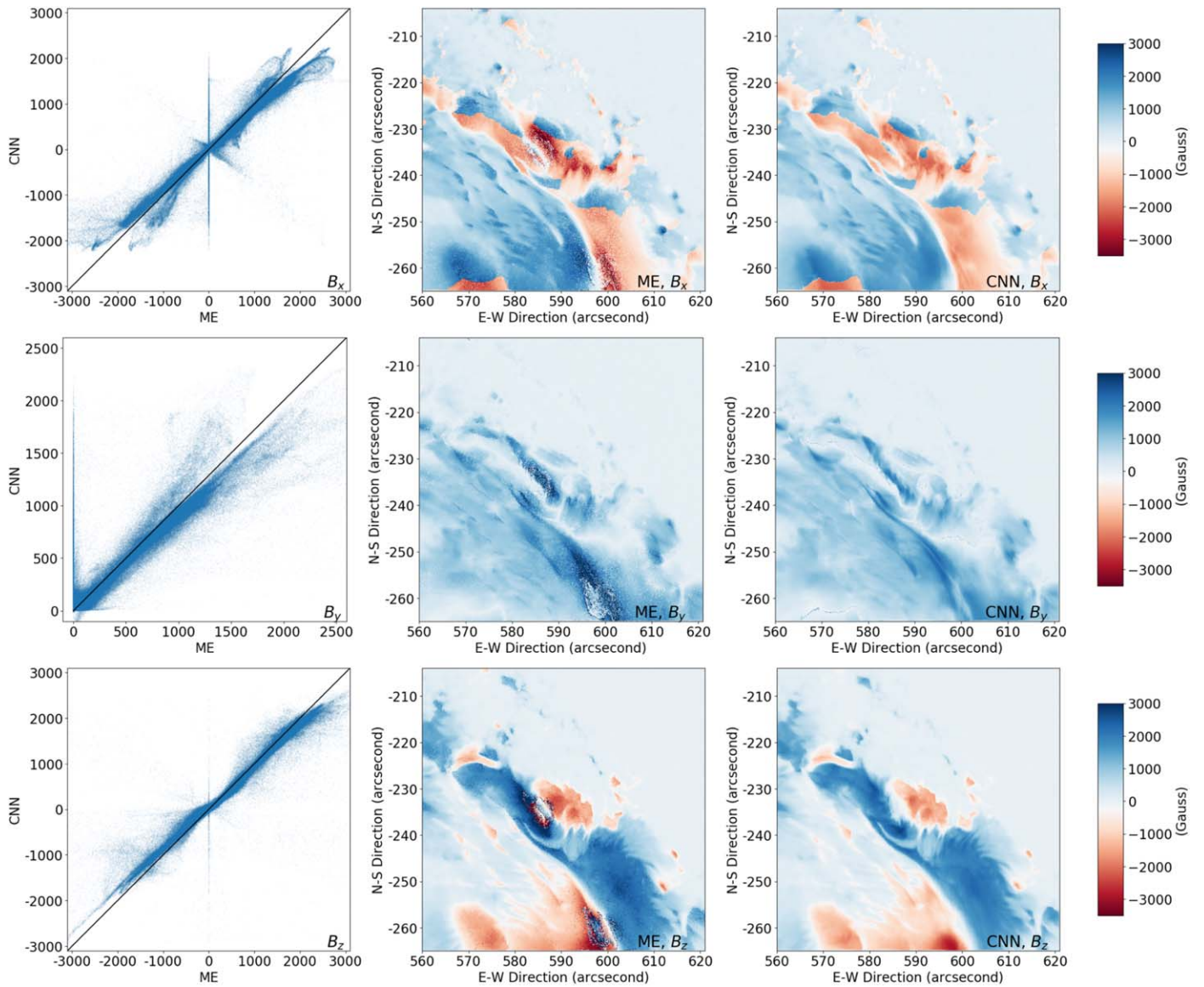
|                    |                                                          | $B_{\text{total}}$ | $B_x$   | $B_y$   | $B_z$   | $\phi$ | $\theta$ |
|--------------------|----------------------------------------------------------|--------------------|---------|---------|---------|--------|----------|
| MAE                | $D_1^{\text{train}} \rightarrow D_4^{\text{test}}$       | 193.680            | 146.010 | 124.783 | 136.892 | 5.497  | 9.009    |
|                    | $D_2^{\text{train}} \rightarrow D_4^{\text{test}}$       | 246.086            | 160.538 | 131.986 | 186.657 | 6.296  | 9.501    |
|                    | $D_3^{\text{train}} \rightarrow D_4^{\text{test}}$       | 231.481            | 153.664 | 129.813 | 173.582 | 5.823  | 7.473    |
|                    | $D_{1,2}^{\text{train}} \rightarrow D_4^{\text{test}}$   | 198.832            | 143.087 | 123.287 | 146.410 | 5.363  | 8.729    |
|                    | $D_{1,3}^{\text{train}} \rightarrow D_4^{\text{test}}$   | 204.086            | 143.244 | 123.685 | 148.227 | 5.284  | 7.925    |
|                    | $D_{2,3}^{\text{train}} \rightarrow D_4^{\text{test}}$   | 201.117            | 137.369 | 119.157 | 162.063 | 5.713  | 7.577    |
|                    | $D_{1,2,3}^{\text{train}} \rightarrow D_4^{\text{test}}$ | 207.075            | 148.718 | 127.467 | 146.775 | 5.674  | 8.679    |
| PA                 | $D_1^{\text{train}} \rightarrow D_4^{\text{test}}$       | 75.0%              | 80.1%   | 86.2%   | 87.2%   | 91.3%  | 79.1%    |
|                    | $D_2^{\text{train}} \rightarrow D_4^{\text{test}}$       | 54.9%              | 77.6%   | 83.7%   | 77.0%   | 87.7%  | 76.3%    |
|                    | $D_3^{\text{train}} \rightarrow D_4^{\text{test}}$       | 71.0%              | 79.0%   | 83.9%   | 81.2%   | 89.5%  | 86.2%    |
|                    | $D_{1,2}^{\text{train}} \rightarrow D_4^{\text{test}}$   | 72.9%              | 81.5%   | 86.2%   | 84.4%   | 91.0%  | 79.9%    |
|                    | $D_{1,3}^{\text{train}} \rightarrow D_4^{\text{test}}$   | 67.8%              | 80.9%   | 85.6%   | 82.9%   | 91.3%  | 82.2%    |
|                    | $D_{2,3}^{\text{train}} \rightarrow D_4^{\text{test}}$   | 72.6%              | 82.8%   | 87.2%   | 83.3%   | 88.7%  | 84.6%    |
|                    | $D_{1,2,3}^{\text{train}} \rightarrow D_4^{\text{test}}$ | 70.9%              | 80.0%   | 84.8%   | 84.1%   | 90.7%  | 79.5%    |
| $R$ -squared       | $D_1^{\text{train}} \rightarrow D_4^{\text{test}}$       | 0.841              | 0.884   | 0.777   | 0.736   | 0.776  | 0.807    |
|                    | $D_2^{\text{train}} \rightarrow D_4^{\text{test}}$       | 0.805              | 0.876   | 0.808   | 0.710   | 0.770  | 0.794    |
|                    | $D_3^{\text{train}} \rightarrow D_4^{\text{test}}$       | 0.769              | 0.867   | 0.763   | 0.687   | 0.785  | 0.824    |
|                    | $D_{1,2}^{\text{train}} \rightarrow D_4^{\text{test}}$   | 0.843              | 0.882   | 0.797   | 0.731   | 0.776  | 0.819    |
|                    | $D_{1,3}^{\text{train}} \rightarrow D_4^{\text{test}}$   | 0.832              | 0.881   | 0.781   | 0.733   | 0.782  | 0.834    |
|                    | $D_{2,3}^{\text{train}} \rightarrow D_4^{\text{test}}$   | 0.835              | 0.894   | 0.788   | 0.714   | 0.780  | 0.821    |
|                    | $D_{1,2,3}^{\text{train}} \rightarrow D_4^{\text{test}}$ | 0.830              | 0.875   | 0.796   | 0.738   | 0.782  | 0.822    |
| PPMCC              | $D_1^{\text{train}} \rightarrow D_4^{\text{test}}$       | 0.936              | 0.943   | 0.888   | 0.859   | 0.881  | 0.902    |
|                    | $D_2^{\text{train}} \rightarrow D_4^{\text{test}}$       | 0.927              | 0.939   | 0.904   | 0.862   | 0.882  | 0.895    |
|                    | $D_3^{\text{train}} \rightarrow D_4^{\text{test}}$       | 0.896              | 0.935   | 0.877   | 0.834   | 0.889  | 0.911    |
|                    | $D_{1,2}^{\text{train}} \rightarrow D_4^{\text{test}}$   | 0.937              | 0.941   | 0.897   | 0.858   | 0.882  | 0.907    |
|                    | $D_{1,3}^{\text{train}} \rightarrow D_4^{\text{test}}$   | 0.934              | 0.942   | 0.891   | 0.861   | 0.885  | 0.915    |
|                    | $D_{2,3}^{\text{train}} \rightarrow D_4^{\text{test}}$   | 0.928              | 0.946   | 0.889   | 0.853   | 0.888  | 0.909    |
|                    | $D_{1,2,3}^{\text{train}} \rightarrow D_4^{\text{test}}$ | 0.933              | 0.940   | 0.899   | 0.863   | 0.885  | 0.909    |
| Outlier difference | $D_1^{\text{train}} \rightarrow D_4^{\text{test}}$       | 19651              | 22317   | 16592   | 12950   | 21951  | 14265    |
|                    | $D_2^{\text{train}} \rightarrow D_4^{\text{test}}$       | 19562              | 22361   | 16772   | 12988   | 21959  | 13705    |
|                    | $D_3^{\text{train}} \rightarrow D_4^{\text{test}}$       | 19647              | 22125   | 16731   | 12956   | 21931  | 15124    |
|                    | $D_{1,2}^{\text{train}} \rightarrow D_4^{\text{test}}$   | 19622              | 22333   | 16645   | 12922   | 21955  | 14305    |
|                    | $D_{1,3}^{\text{train}} \rightarrow D_4^{\text{test}}$   | 19650              | 22277   | 16573   | 12967   | 21961  | 13425    |
|                    | $D_{2,3}^{\text{train}} \rightarrow D_4^{\text{test}}$   | 19691              | 22072   | 16668   | 13004   | 21949  | 15841    |
|                    | $D_{1,2,3}^{\text{train}} \rightarrow D_4^{\text{test}}$ | 19660              | 22313   | 16594   | 12970   | 21954  | 13645    |



**Figure A1.** Comparison between the ME and CNN methods for deriving  $B_x$ ,  $B_y$ , and  $B_z$  based on the test image from AR 12371 collected on 2015 June 25 20:00:00 UT, where training data were taken from AR 12371 on 2015 June 22. Displayed from top to bottom are the results for  $B_x$ ,  $B_y$ , and  $B_z$ . The first column shows scatter plots where the X-axis and Y-axis represent the values obtained by the ME and CNN methods, respectively. The black diagonal line in each scatter plot corresponds to pixels whose ME-calculated values are identical to CNN-inferred values. The second column shows magnetic maps derived by the ME method. The third column shows magnetic maps inferred by our CNN method.



**Figure A2.** Comparison between the ME and CNN methods for deriving  $B_x$ ,  $B_y$ , and  $B_z$  based on the test image from AR 12665 collected on 2017 July 13 18:35:00 UT, where training data were taken from AR 12371 on 2015 June 22. Displayed from top to bottom are the results for  $B_x$ ,  $B_y$ , and  $B_z$ . The first column shows scatter plots where the X-axis and Y-axis represent the values obtained by the ME and CNN methods, respectively. The black diagonal line in each scatter plot corresponds to pixels whose ME-calculated values are identical to CNN-inferred values. The second column shows magnetic maps derived by the ME method. The third column shows magnetic maps inferred by our CNN method.



**Figure A3.** Comparison between the ME and CNN methods for deriving  $B_x$ ,  $B_y$ , and  $B_z$  based on the test image from AR 12673 collected on 2017 September 6 19:18:00 UT, where training data were taken from AR 12371 on 2015 June 22. Displayed from top to bottom are the results for  $B_x$ ,  $B_y$ , and  $B_z$ . The first column shows scatter plots where the X-axis and Y-axis represent the values obtained by the ME and CNN methods, respectively. The black diagonal line in each scatter plot corresponds to pixels whose ME-calculated values are identical to CNN-inferred values. The second column shows magnetic maps derived by the ME method. The third column shows magnetic maps inferred by our CNN method.

### ORCID iDs

Jiasheng Wang <https://orcid.org/0000-0001-5099-8209>  
 Jason T. L. Wang <https://orcid.org/0000-0002-2486-1097>  
 Haimin Wang <https://orcid.org/0000-0002-5233-565X>

### References

- Asensio Ramos, A., & de la Cruz Rodríguez, J. 2015, *A&A*, **577**, A140  
 Asensio Ramos, A., & Díaz Baso, C. J. 2019, *A&A*, **626**, A102  
 Auer, L. H., Heasley, J. N., & House, L. L. 1977, *SoPh*, **55**, 47  
 Borrero, J. M., Tomczyk, S., Kubo, M., et al. 2011, *SoPh*, **273**, 267  
 Cao, W., Goode, P. R., Ahn, K., et al. 2012, in ASP Conf. Ser. 463, NIRIS: The Second Generation Near-Infrared Imaging Spectro-polarimeter for the 1.6 Meter New Solar Telescope, ed. T. R. Rimmele et al. (San Francisco, CA: ASP), 291  
 Cao, W., Jing, J., Ma, J., et al. 2006, *PASP*, **118**, 838  
 Carroll, T. A., & Kopf, M. 2008, *A&A*, **481**, L37  
 Carroll, T. A., & Staude, J. 2001, *A&A*, **378**, 316  
 Collados, M. 2008, *Proc. SPIE*, **7012**, 70120J  
 del Toro Iniesta, J. C., & Ruiz Cobo, B. 1996, *SoPh*, **164**, 169  
 Frutiger, C., Solanki, S. K., Fligge, M., & Bruls, J. H. M. J. 2000, *A&A*, **358**, 1109  
 Galton, F. 1886, *The Journal of The Anthropological Institute of Great Britain and Ireland*, **15**, 246  
 Goode, P. R., & Cao, W. 2012, in ASP Conf. Ser. 463, The 1.6 m Off-Axis New Solar Telescope (NST) in Big Bear, ed. T. R. Rimmele (San Francisco, CA: ASP), 357  
 Goodfellow, I. J., Bengio, Y., & Courville, A. C. 2016, Deep Learning, Adaptive Computation and Machine Learning (Cambridge, MA: MIT Press), <http://www.deeplearningbook.org/>  
 Lagg, A., Woch, J., Krupp, N., & Solanki, S. K. 2004, *A&A*, **414**, 1109  
 Landi Degl'Innocenti, E. 1984, *SoPh*, **91**, 1  
 Landolfi, M., Landi Degl'Innocenti, E., & Arena, P. 1984, *SoPh*, **93**, 269  
 LeCun, Y., Bengio, Y., & Hinton, G. 2015, *Natur*, **521**, 436  
 Liu, C., Cao, W., Chae, J., et al. 2018, *ApJ*, **869**, 21  
 McHugh, M. L. 2012, *Biochimica Medica*, **22**, 276  
 McMullin, J. P., Rimmele, T. R., Keil, S. L., et al. 2012, *Proc. SPIE*, **8444**, 844407

- Pearson, K. 1895, [RSPS](#), **58**, 240
- Press, W. H., Flannery, B. P., Teukolsky, S. A., & Vetterling, W. T. 1991, [MaCom](#), **56**, 396
- Quintero Noda, C., Asensio Ramos, A., Orozco Suárez, D., & Ruiz Cobo, B. 2015, [A&A](#), **579**, A3
- Rees, D., Guo, Y., López Ariste, A., & Graham, J. 2004, Lecture Notes in Computer Science (Berlin: Springer), 388
- Ruiz Cobo, B., & Asensio Ramos, A. 2012, [A&A](#), **549**, L4
- Ruiz Cobo, B., & del Toro Iniesta, J. C. 1992, [ApJ](#), **398**, 375
- Sen, A., & Srivastava, M. 1990, Regression Analysis (New York: Springer)
- Skumanich, A., & Lites, B. W. 1987, [ApJ](#), **322**, 473
- Socas-Navarro, H. 2003, [NN](#), **16**, 355
- Socas-Navarro, H. 2005, [ApJ](#), **621**, 545
- Socas-Navarro, H., López Ariste, A., & Lites, B. W. 2001, [ApJ](#), **553**, 949
- Teng, F. 2015, [SoPh](#), **290**, 2693
- Unno, W. 1956, PASJ, **8**, 108
- Wang, H., Cao, W., Liu, C., et al. 2015, [NatCo](#), **6**, 7008
- Wang, H., Liu, C., Ahn, K., et al. 2017, [NatAs](#), **1**, 85
- Xu, Y., Cao, W., Ahn, K., et al. 2018, [NatCo](#), **9**, 46
- Xu, Y., Cao, W., Ding, M., et al. 2016, [ApJ](#), **819**, 89