The Astrophysical Journal, 999:171 (16pp), 2026 March 10 https://doi.org/10.3847/1538-4357/ae3f97

Deep-learning-based Identification of Solar Magnetic Tornadoes and Their Spatial Properties during Solar Minimum and Maximum

Mark I. Blumenau[1,2], Olga Khabarova[3], Ilia S. Nikitin[1,2], and Dmitrii L. Vorobev[1,4]
[1] Pushkov Institute of Terrestrial Magnetism, Ionosphere and Radio Wave Propagation of the Russian Academy of Sciences (IZMIRAN), 108840 Moscow, Russia; mblumenau@hse.ru
[2] HSE University, 101000 Moscow, Russia
[3] Interdisciplinary Centre for Security, Reliability and Trust (SnT), University of Luxembourg, L-1855 Luxembourg, Luxembourg
[4] Department of Physics, Carnegie Mellon University, Pittsburgh, PA 15213, USA
Received 2025 August 22; revised 2025 December 24; accepted 2026 January 19; published 2026 March 2

Abstract

Solar magnetic tornadoes are dynamic, spiral-shaped plasma structures characterized by helical magnetic fields and rotating plasma flows in the solar atmosphere. They play a significant role in the transport of energy and mass within the solar environment. Identifying and analyzing solar magnetic tornadoes is challenging due to their transient nature and complex morphology and the large volume of associated observational data. We propose two automated methods for detecting these magnetoplasma structures using modern deep learning techniques. Our models search for twisted prominences in the solar corona visible at the solar limb. Our approach involves analyzing the Solar Dynamics Observatory Atmospheric Imaging Assembly 171 Å images using convolutional and recurrent neural networks. By applying established techniques, the methods proposed can detect previously unknown magnetic tornadoes alongside those documented in the literature. The models are trained on 10,294 instances, which corresponds to detection of ∼100 tornadoes with high precision and recall. Identification of 1,476,885 new instances is performed. The resulting database allows for the first comparative analysis of magnetic tornadoes' spatial distributions across solar cycle phases. We find that tornadoes can serve as tracers of environments prone to reconnection. During solar minimum, these structures occur at the boundaries of coronal holes with strong current sheets and at the edges of polar conic current sheets. During solar maximum, they appear at the footpoints of magnetic loops and are associated with polarity inversion lines.

Unified Astronomy Thesaurus concepts: Solar atmosphere (1477); Solar prominences (1519); Interdisciplinary astronomy (804); Computational astronomy (293)

Materials only available in the online version of record: animations

1. Introduction

Solar magnetic tornadoes are often treated as specific twisting filaments or tornado-like prominences (S. Wedemeyer-Böhm et al. 2012; Y. Su et al. 2014; S. Wedemeyer & O. Steiner 2014). These are magnetoplasma structures with signatures of vortex flows rooting in the photosphere and expanding into the solar corona. The plasma within the tornado is highly dynamic, with strong shear flows and turbulence driven by the twisting of the magnetic field lines. According to statistics (S. Wedemeyer et al. 2013), such plasma objects can extend up to 140″ above the photosphere and live up to 200 hr. The temperature of tornadoes is supposed to be lower than that in the surrounding plasma, although some studies show that the rotating structures can contain both hot and cool plasma (Y. Su et al. 2014). Solar magnetic tornadoes resemble terrestrial tornadoes; however, they differ significantly from those, since their dynamics is described by the magnetohydrodynamics equations. E. Pettit (1943) was the first to suggest that spiral prominences appeared on the Sun in 1932, but most studies of tornadoes on the Sun date back to the last decade.

The nature of the formation of such tornadoes is still under debate, and it is not yet clear how the twisting of magnetic field lines leads to the formation of a rotating column of plasma or how magnetic tornadoes interact with the surrounding solar atmosphere. There are studies suggesting that the tornado-like bushy top part of some prominences just resembles a tornado because it is formed by numerous threads with counterstreaming flows, and it does not rotate, but rather shakes or oscillates (B. Schmieder et al. 2017; K. Barczynski et al. 2021), while other studies claim the rotation of both the magnetic field and plasma in magnetic tornadoes (Y. Su et al. 2012, 2014; S. Wedemeyer et al. 2013). Anyway, the existence of such structures is not debatable anymore, but the nature of the phenomenon requires further investigation. Despite the complexity of the dynamics of these plasma objects, the possibility of modeling the formation and development of tornadoes on the Sun has been proved (S. Wedemeyer-Böhm et al. 2012; M. Luna et al. 2015). The existence of tornadoes is supposed to explain one of the main mysteries of solar physics, namely, why the corona, the outermost layer of the solar atmosphere, is much hotter than all other layers (S. Wedemeyer-Böhm et al. 2012; H. Kuniyoshi et al. 2023).

Over the past decade, many examples of magnetic tornadoes have been found (X. Li et al. 2012; O. Panasenco et al. 2013; N. K. Panesar et al. 2013; S. Wedemeyer et al. 2013; Y. Su et al. 2014; H. Chen et al. 2017; I. Mghebrishvili et al. 2018). They were visually identified and individually analyzed, which made statistical studies time consuming. The reason for this is that their morphological diversity does not allow for an absolutely precise definition of the phenomenon, making it

difficult to analyze these structures with computer methods of image processing. K. Tziotziou et al. (2023), in their recent review, suggest a classification of vortex motions recognized as tornadoes in the literature and distinguish four main types of them. The common point between them all is an obvious manifestation of vortical motions or the impression of rotation visible by eye. Numerous plasma objects, different by nature, fall into this classification. Meanwhile, automation of magnetic tornado detection is a necessary step for the qualitative and quantitative study of the phenomenon even if some tornado-like structures are not rotating columns of plasma.

The use of neural networks for recognition of solar magnetic tornadoes could have significant implications for the understanding of these structures by the community of space physicists. It would allow us to identify and study magnetic tornadoes in large datasets of solar images, which would be difficult and time-consuming to analyze manually. The easy finding of magnetic tornadoes at the solar limb could help us analyze their properties and behavior not previously recognized. Neural networks were employed in astrophysics shortly after they had first been developed (A. Miller 1993; A. Asensio Ramos et al. 2023). In particular, they have been successfully used to detect coronal holes (R. Jarolim et al. 2021) and solar flares (R. A. Fernandez Borda et al. 2002). An overview classification of solar images shows the importance of the application of neural networks and other automatic methods of object recognition to solar physics with an exponentially growing amount of data (I. F. Scholl & S. R. Habbal 2007; A. Hamada et al. 2018; J. A. Armstrong & L. Fletcher 2019; E. Illarionov et al. 2020; J. A. Linker et al. 2021; J. Liu et al. 2024).

Below, we show a possibility of identifying solar magnetic tornadoes with machine learning methods, discuss the details of using neural networks, and show examples of the magnetic tornadoes found automatically. In this study, we consider only stand-alone structures seen at the limb and propose to use (1) a newly created neural network with convolutional, recurrent, and fully connected layers and (2) a well-known You Only Look Once (YOLO) network (J. Redmon et al. 2016). We train the networks using a dataset of high-resolution solar images previously verified to contain magnetic tornadoes, enabling them to learn the characteristic patterns of these structures. Once trained, the networks analyze new solar images to identify magnetic tornadoes with high accuracy.

Further research is required to fully understand the nature of magnetic tornadoes and their role within the complex solar atmosphere. In the meantime, the development of automated identification methods for magnetic tornadoes can facilitate more efficient and accurate analysis of other complex astrophysical phenomena. The results of this study may potentially enable differentiation between various types of magnetic tornadoes. With rapid automated detection, comprehensive statistical analyses of their spatial and temporal evolution will become feasible, allowing the discovery of previously hidden patterns in their interactions with other magnetoplasma structures and their connection to eruptive processes.

2. Methods and Data

2.1. Main Ideas

Solar plasma structures of different morphology can hypothetically be distinguished by various computer methods of image processing. In this work, we focus on neural networks, which are a subclass of machine learning methods used to solve many problems (L. Alzubaidi et al. 2021). One of the examples is a combination of convolutions. For example, let us take the limb image from 2011 September 25, obtained using the Atmospheric Imaging Assembly (AIA) instrument on board the Solar Dynamics Observatory (SDO) in line 171 Å[5] as shown in Figure 1(a). One can apply various Sobel filters (I. Sobel 2014) to it to obtain Figures 1(b), (c), and (d). The filter in Figure 1(b) reveals mostly horizontal directions, that in Figure 1(c) shows vertical patterns, and in Figure 1(d), it highlights features directed at a 45° angle. As a result, objects possessing different morphological properties look different. The approach with trainable convolution layers, in which the filters are learned via gradient descent, has been used in CHRONNOS, where a neural network identifies coronal holes (R. Jarolim et al. 2021). Similarly, T. Zhang et al. (2024) employ trainable convolutions in their study of solar prominences. Following this logic, and given that we are interested in dynamic objects, we combine convolutions with recurrent neural layers, since they are designed to process sequences of data, including time series and natural language (human spoken and written languages). Examples of these layers used can be found in R. Jozefowicz et al. (2015), B. Lim & S. Zohren (2021), and S. Mohsen (2023).

In some cases, if the object under study does not have pronounced temporal dynamics or differs significantly from background and other structures, neural networks from the YOLO family can be used (A. Al-Owais et al. 2023; K. Grishin et al. 2023; Y. Wan & J. Li 2023). Such networks have been employed, for example, for solar active region detection (L. Quan et al. 2021). They usually work on single frames, without the need to take into account information from previous ones. The main advantage of this method is the ability to extract the dimensions of objects of interest. However, in this case, there is a risk of misidentifying other structures or missing relevant objects if they are very similar to the studied ones or change greatly over time.

2.2. Data and Materials

In our study, we use solar images obtained at wavelength 171 Å from the AIA instrument on board the SDO, downloading them from the Joint Science Operations Center (JSOC) database (J. R. Lemen et al. 2012; W. D. Pesnell et al. 2012). The time cadence between images is 6 minutes. The image of the Sun obtained from the SDO AIA data is shown in the upper left corner of Figure 2(a). The JSOC images themselves are black and white, but we use the standard processing and color maps to show the colored version. We also perform basic checks, i.e., verifying that the Sun is in the frame and that the mean brightness of the images is not too high or too low. The down arrow indicates the transformations associated with image cropping and alignment. The result is a limb image only, without the solar disk visible. For both approaches, the original data have a resolution of 4096 by 4096 pixels, and the size of the cut limb is about 12,500 by 384 pixels. The choice of 384 pixels is motivated by the measured solar radius (~1583 pixels for 2020 May and 2024 May) and represents a compromise between maintaining a sufficient safety margin and ensuring complete coverage of solar tornadoes. The size of the limb depends on the data inside

[5] See https://sdo.gsfc.nasa.gov/.

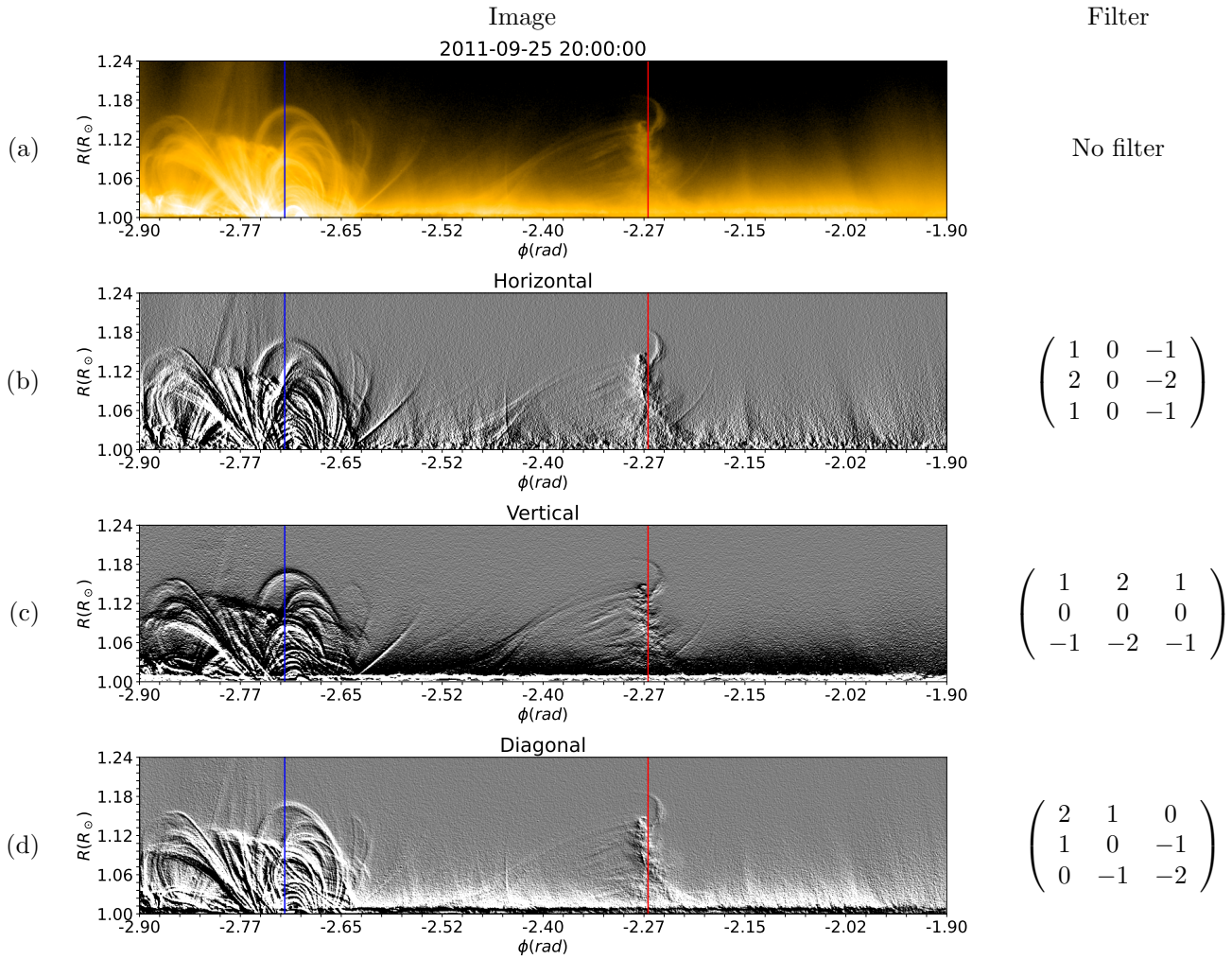


Figure 1. Example of filter use for identification of differently directed objects in the solar atmosphere. SDO AIA 171 Å data. (a) Initial image. (b) Sobel filter, x-component. (c) Sobel filter, y-component. (d) Modified Sobel filter. The red vertical line indicates the solar tornado, and the blue line indicates another object.

the fits file. The size of the images used for neural network predictions depends on the approach and will be described below.

To train the neural network, we used a dataset containing some magnetic tornadoes previously identified in the studies by X. Li et al. (2012), O. Panasenco et al. (2013), N. K. Panesar et al. (2013), S. Wedemeyer et al. (2013), Y. Su et al. (2014), and I. Mghebrishvili et al. (2018). Additionally, we manually identified more tornadoes through visual inspection. This search was conducted using AIA 171 Å data from the same data source.[6] Figure 3 shows four example images of solar magnetic tornadoes previously discussed in the literature and used in our study to develop automated identification methods. In 171 Å images, solar magnetic tornadoes typically appear as distinct dark structures near the limb, exhibiting elongated lower bodies with broader, bush-like tops. The rotational nature of these features becomes evident when analyzing image sequences. Tornadoes often occur in clusters or chains. Figures 3(a) and (b) display multiple long-lived tornadoes described by S. Wedemeyer et al. (2013), while Figures 3(c) and (d) present individual examples identified by X. Li et al. (2012), N. K. Panesar et al. (2013), and Y. Su et al. (2014).

The total number of tornadoes in the dataset is about 100. Videos in the training and test samples are divided so that they contain tornadoes of different dates. The test set includes only tornadoes from 2021, while the training set includes images from 2011. The exact number of images in the dataset depends on the approach and is provided below. The training is carried out on either an Nvidia GeForce RTX 3060 GPU or an Nvidia V100 GPU. The source code is written in Python, and the PyTorch library is used for all neural networks (A. Paszke et al. 2019).

To compare the statistical results describing the spatial characteristics of magnetic tornadoes with the solar magnetic field topology, we use the results of magnetic field line reconstruction in the solar corona via the potential field source surface (PFSS) model provided by the Lockheed Martin Solar and Astrophysics Laboratory.[7] The corresponding flux-transport model employs photospheric magnetograms, with the algorithm described by C. J. Schrijver & M. L. DeRosa (2003).

[6] The complete dataset is hosted on a MiniO instance at https://minio.smtornado.com/minio/ui/browser/datasets and can be downloaded as a zip archive.

[7] The Interactive Solar Fieldline Viewer (https://sdowww.lmsal.com/sdomedia/SunInTime/webgl_tool/sv4/).

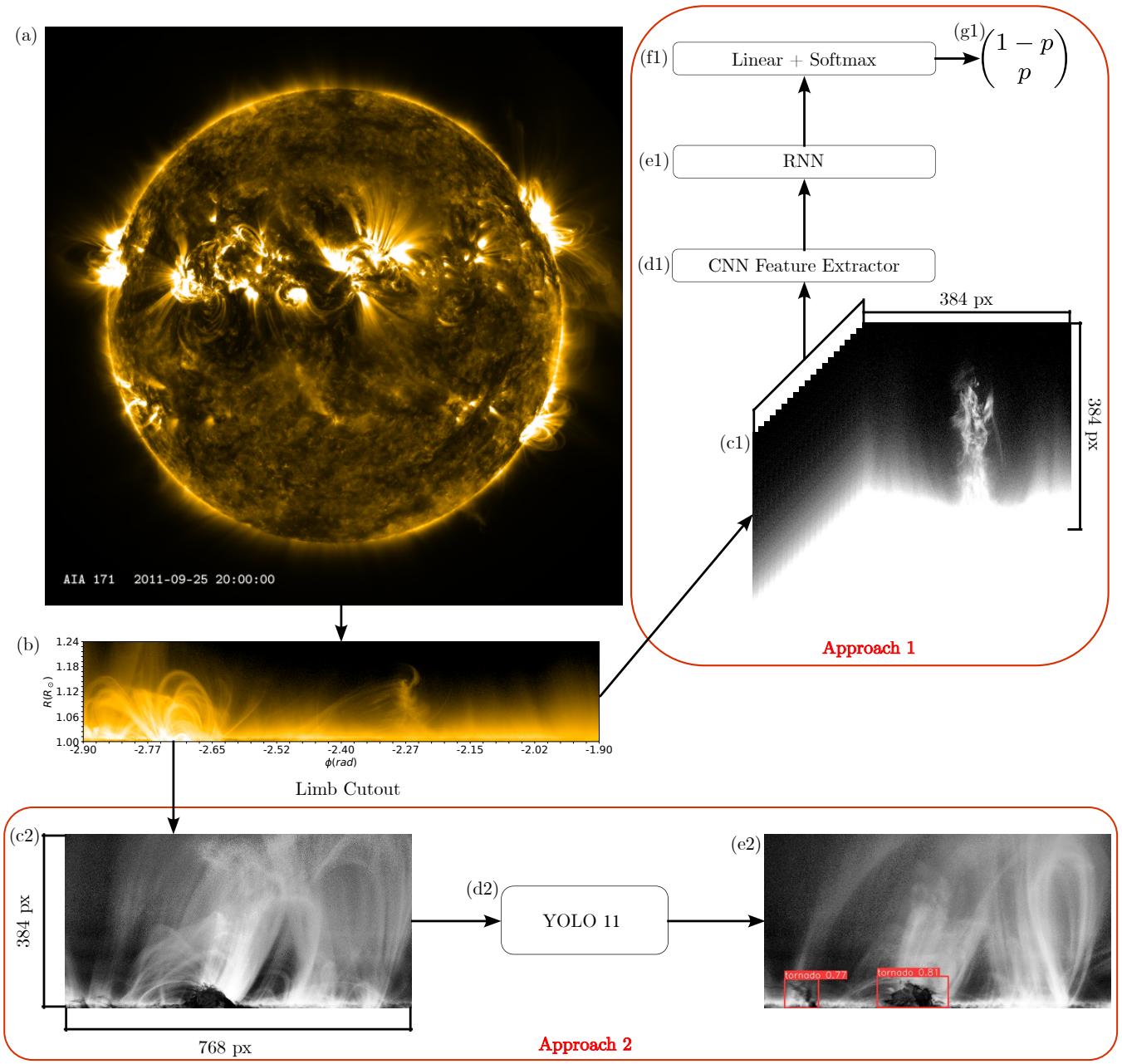
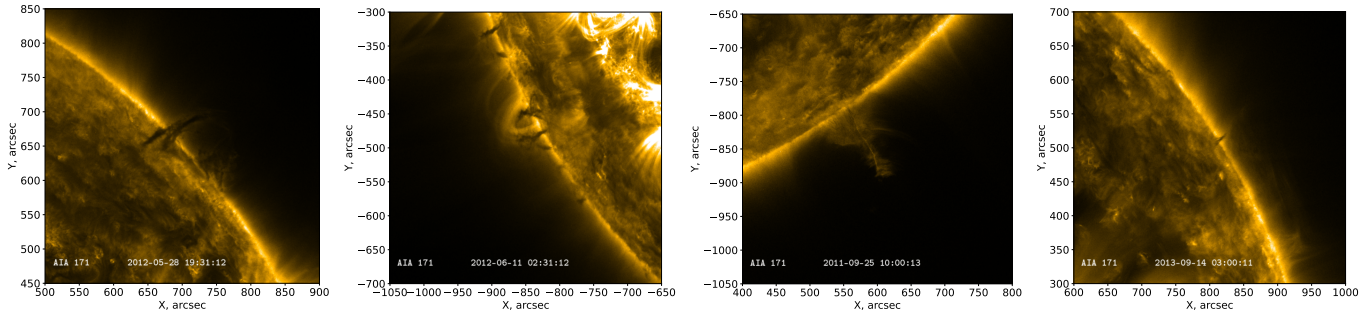


Figure 2. Schematic overview of the two data processing pipelines used for automated identification of solar magnetic tornadoes. (a) Original AIA 171 Å image of the full solar disk. (b) Cropped solar limb (≈12,500 × 384 pixels). Approach 1 includes the following steps: (c1) construction of image sequences consisting of 20 consecutive frames of the same limb segment (384 × 384 pixels) with a temporal cadence of 6 minutes; (d1) feature extraction, in which each frame is transformed into a fixed-length feature vector (256 or 512 elements, depending on the network architecture); (e1) aggregation of the feature vectors into a single representation using an RNN; (f1) application of a fully connected layer with softmax activation to estimate the probability of tornado presence; and (g1) output probability vector indicating the presence or absence of a tornado within the image sequence. Approach 2 operates on individual images: (c2) a single limb patch is processed independently (without temporal sequencing), (d2) feature processing and object detection using the YOLO 11 architecture on resized patches, and (e2) output bounding boxes with corresponding coordinates and confidence scores for detected tornadoes.

2.3. Pipelines

We present below two different approaches that can be used to automatically detect prominent structures in the solar atmosphere, such as magnetic tornadoes. Depending on the approach (pipeline), the output of the model can be either the probability of the presence of a tornado in a given set of images (approach 1) or a set of initial images supplemented by a set of rectangles highlighting solar magnetic tornadoes and showing numbers corresponding to the model's confidence that there is a tornado inside a given rectangle (approach 2). The YOLO model is the basis of approach 2. Both methods have their pros and cons. YOLO is a well-known and easy-to-use neural network. Meanwhile, despite all the convenience of approach 2, YOLO does not have a recurrent neural network (RNN) module, so the rotation of a tornado cannot be taken into account without add-ons. Since YOLO is known to be the model that successfully recognizes objects in the solar atmosphere (L. Quan et al. 2021), its application to the particular task seems to be reasonable. In this study, we show its ability to successfully recognize solar magnetic tornadoes in a convenient way, which can potentially make it a useful and



(a) (b) (c) (d)

Figure 3. Examples of magnetic tornadoes discussed in the literature. Tornadoes shown in panels (a) and (b) were studied by S. Wedemeyer et al. (2013), that in panel (c) by X. Li et al. (2012) and N. K. Panesar et al. (2013), and tornado in panel (d) by Y. Su et al. (2014). The corresponding heights above the limb are approximately 45,000, 15,000, 75,000, and 15,000 km, measured using Helioviewer.

easily reachable tool helping the plasma community to study the properties of these specific prominences.

2.3.1. Approach 1

The pipeline for this approach is schematically represented in Figures 2(d1)–(f1). The arrow to the right of Figure 2(b) shows the cutting of a large image of the limb into smaller ones and the formation of sequences. The sequence is a set of 20 frames of the same segment of the limb with a time difference between each pair of 6 minutes. Each frame has a size of 384 by 384 pixels. Patches overlap to reduce border effects during classification. The amount of overlap depends on the exact width of the limb cutout. Because the limb cutout represents a circular structure, overlap is also applied at the beginning and end of the ∼12,500 by 384 image. Each patch sequence is classified independently, and no additional postprocessing is performed. In total, the whole limb cutout is sliced into 48 patches. The resulting training set consists of 4693 videos, 2964 of which contain tornadoes, while the test set includes 125 videos, 63 of which contain objects under study. Although solar tornadoes are relatively rare, the class imbalance is reduced by preparing fewer nontornado videos for training. The training and validation dataset preparation ensures that the patching process does not split tornadoes in half. The video made of images selected is classified by the computer vision model.

To solve the classification problem, we propose to use a combination of several modules of various types. The first module, called the feature extractor, is responsible for processing each image in the sequence and converting it into a vector of a predetermined length. This vector contains the most important information for the neural network from the frame. We used MobileNetV3 (A. Howard et al. 2019) as the feature extractor because of its comparatively low computation cost. The corresponding length of the output vectors was 256 or 512.

The basis of the used feature extractor is various blocks. For the sake of completeness, we briefly summarize the main blocks of the MobileNetV3 architecture in Appendix A.

After images pass through the feature extractor, the resulting combination of vectors, the number of which corresponds to the number of frames in the set, is combined into a matrix (stack) and transmitted to the RNN. In them, the elements form a directed sequence, which makes it possible to take into account the ordering of the frames in time. We have tested gated recurrent unit (GRU; K. Cho et al. 2014) and long short-term memory (LSTM; S. Hochreiter & J. Schmidhuber 1997) layers widely used in natural language processing tasks. All of them are described in Appendix A. The result is another vector processed by the fully connected layer (D. E. Rumelhart et al. 1986).

The output of the fully connected layer is a vector of length 2. To get the probabilities, the softmax function (D. E. Rumelhart et al. 1986) is applied to it.

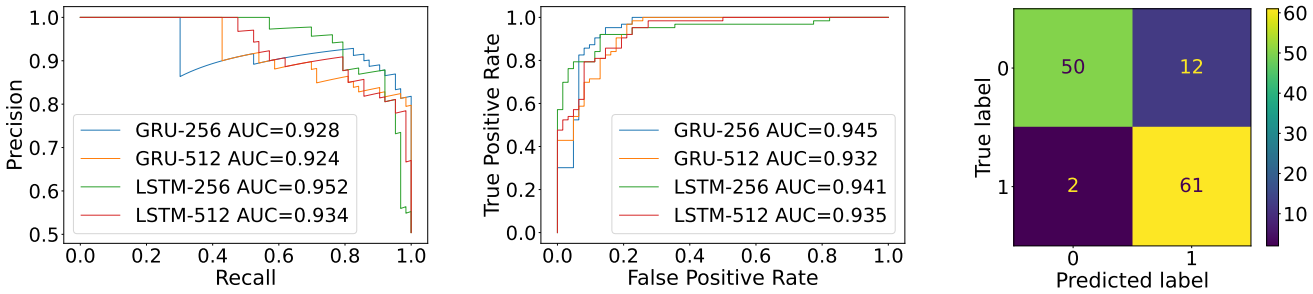
After that, the second element of the vector is taken, which is the probability of having a tornado on a sequence of frames.

For training the neural network, in addition to preprocessed data, it is required to introduce a loss function that characterizes how wrong the model can be. An optimization method that implements the selection of parameters (matrices of weights and filters) should also be used. We have chosen the cross-entropy loss function (C. E. Shannon 1948) widely used for solving the classification problem.

The optimizer used is the Adam algorithm (D. P. Kingma & J. Ba 2014; see Appendix A for description). We used the following default PyTorch parameter values: $m_0 = 0$, $v_0 = 0$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\gamma = 0.001$. Batch size was selected to maximize GPU memory utilization. Input data were normalized, but no augmentations were used for training.

2.3.2. Approach 2

This method offers the greatest ease of use, as it relies on well-established community-developed libraries, most notably Ultralytics (G. Jocher et al. 2023), PyTorch (A. Paszke et al. 2019), and OpenCV (G. Bradski 2000). When applied to the particular problem, a potential limitation of the approach is its inability to account for the rotational dynamics of the structures, as the model operates on individual frames. This is not a significant issue when the magnetic tornado is bright and isolated from overlapping features such as coronal loops. However, the model's performance in scenarios involving multiple or less distinct tornado-like structures remains uncertain. Given this characteristic of the YOLO framework, it is best suited for detecting the most prominent and well-defined cases—similar to previous case studies that focused exclusively on clearly identifiable events while excluding more complex instances. Expert validation of the model's outputs may still be necessary, but its ease of use and efficiency make it a practical tool despite these limitations. Additional advantages include the widespread adoption of YOLO, its



(a) (b) (c)

Figure 4. Key characterization curves for the models of the best performance. (a) PR curves. (b) ROC curves. (c) Confusion matrix for MobileNetV3 with output vector size 256 and GRU layers.

accessibility, and the extensive body of literature supporting its application.

Using this architecture, image processing is conducted on a frame-by-frame basis. To satisfy the model's input requirements, images are resized to dimensions divisible by 32; in this case, 768 × 384 pixels are used. Similarly to the first approach, the images overlap. In total, each initial limb cutout is divided into 20 patches. The resulting training set consists of 6332 images containing 9226 bounding boxes, while the test set includes 930 images with 1068 bounding boxes. The model employed is YOLO 11n from the Ultralytics library (G. Jocher et al. 2023). This model outputs the coordinates of the tornado center along with the width and height of a bounding box that encloses the structure. In addition to the bounding box, it provides a confidence score, indicating the model's certainty in detecting a given structure.

A brief overview of the YOLO v8 architecture is available.[8] The YOLO 11 architecture, while similar, includes some modifications and is documented by Ultralytics. Both versions are primarily composed of the blocks and layers described in Appendix A, with minor structural adjustments. All training parameters (including augmentations) were kept at their default values for Ultralytics version 8.3.70.

3. Results

3.1. Approach 1

To evaluate our model, we introduce below the key metrics shown in Figure 4, which are the area under curve (AUC) precision recall (PR; M. Sokolova & G. Lapalme 2009; D. M. Powers 2020), the AUC receiver operating characteristic (ROC; M. Sokolova & G. Lapalme 2009; D. M. Powers 2020), the precision, and the recall. The AUC is calculated numerically for the PR and ROC curves. The PR curve is drawn in recall and precision coordinates. ROC is plotted in false-positive rate (FPR) and true-positive rate (TPR) coordinates. We do not rely on the accuracy metric alone to select the best model, since it generally fails to distinguish between different types of errors or reflect class-specific performance. Models with fundamentally different behavior can achieve similar accuracy values, making the metric insufficient as a stand-alone criterion for model selection. In our case, a random model would get close to 0.5 on our validation set. Let us define all the terms listed above:

$$\mathrm{TPR} = \mathrm{Recall} \stackrel{\mathrm{def}}{=} \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}}, \quad (1)$$

$$\mathrm{FPR} \stackrel{\mathrm{def}}{=} \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}}, \quad (2)$$

$$\mathrm{Precision} \stackrel{\mathrm{def}}{=} \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}, \quad (3)$$

$$\mathrm{Accuracy} \stackrel{\mathrm{def}}{=} \frac{\text{True Positive} + \text{True Negative}}{\text{Positive} + \text{Negative}}. \quad (4)$$

The higher the value of each metric, the better the model performs. For all metrics, the best value is 1. AUC-ROC can be thought of as a metric that shows the ability of the model to separate negative and positive classes. If its value is 0.5, it refers to a random classifier. The ROC graph of an ideal classifier passes through the points (0, 0), (0, 1), and (1, 1). A random classifier is characterized by a straight line connecting points (0, 0) and (1, 1).

All metrics are calculated for a set of 20 frames. The same tornado is contained in different sequences of frames shifted relative to each other in time, so there are more video files than individual tornadoes. Table 1 shows the metrics for all tested convolutional neural network (CNN) and RNN combinations.

The precision and recall metrics are calculated for a threshold probability of 0.5, which we consider the best compromise in terms of maintaining high recall, ensuring the model's ability to not miss a tornado, and precision, which characterizes the model's ability to leave only real tornadoes. According to our tests, the best model is a combination of MobileNetV3 with a vector length of 256 and GRU as an RNN. Showing the best results in terms of the metrics AUC-ROC, recall, and accuracy, this network is also the smallest in terms of the number of parameters, which allows one to quickly process data even on weak hardware. Figure 4 shows all PR and ROC curves, as well as the confusion matrix for the selected model.

The confusion matrix in Figure 4(c) consists of four values illustrating the success of the model's predictions. The vertical axis shows the correct answers on the labeled dataset (0—"no tornado" or 1—"there is a tornado"), and the model predictions are given on the horizontal axis.

In object classification problems, various types of errors can occur. A type I error occurs when a false-positive result occurs. A false-negative result is a type II error. The number of correctly identified sequences of frames without tornadoes is

[8] https://github.com/ultralytics/ultralytics/issues/189

Table 1
Results and Metrics

Metric	MobileNetV3 GRU-256	MobileNetV3 GRU-512	MobileNetV3 LSTM-256	MobileNetV3 LSTM-512
AUC-PR	0.928	0.924	0.952	0.934
AUC-ROC	0.945	0.932	0.941	0.935
Recall (@0.5)	0.968	0.841	0.794	0.825
Precision (@0.5)	0.836	0.841	0.926	0.852
Accuracy	0.888	0.84	0.864	0.84
Validation loss	0.44	0.454	0.386	0.508

Note. The table shows the metrics calculated for frame sequences for each model. The closer the metric is to 1, the better, with the only exception being validation loss, where smaller values are better.

displayed in the upper left corner of the confusion matrix. The upper right corner represents the sequences incorrectly classified as containing tornadoes (type I error). Conversely, the lower left corner indicates type II errors, where the model failed to detect tornadoes present in the sequences. Finally, the lower right corner corresponds to correctly identified tornado sequences.

It is important to note that our model has no direct analogs. Consequently, we cannot compare the results, including those shown in the confusion matrix and the curves in Figure 4, with similar findings from previous studies, as is typically done in the scientific literature. The absence of comparable models underscores the novelty of our approach. Future studies will provide a basis for benchmarking our model against subsequent advancements, offering a clearer assessment of its relative effectiveness.[9]

Despite the good quality of the models built, there are magnetic tornadoes that are missed because of the general limitations of neural networks. They are unable to distinguish objects that are too different from those in the training sample. This means that rare structures, unlike most tornadoes, are likely to be missed. Some tornadoes might have been missed or misidentified due to the complex structure of the magnetic field lines along which the plasma flows. For example, if a tornado on the limb is obscured by closed magnetic field lines framing the active region, it cannot be reliably determined. The left panels in Figure 5 show examples of tornado chains detected by our method in the format transmitted to the neural network. The text is added for visualization and is not used for the neural network prediction. On the right, colored panels show the same tornadoes in the generally accepted format. The details of the evolution of these tornadoes on the disk can be seen in the corresponding videos. The sizes of the tornadoes in Figure 5 calculated above the solar limb are about 95,000 and 23,000 km.

To facilitate research for specialists studying magnetic tornadoes, we have compiled a table containing sequences of frames with a high likelihood of tornado presence. The table includes events where the model-predicted probability of a tornado is 0.5 or higher. However, users have the flexibility to adjust this probability threshold to suit the requirements of their specific research objectives, enabling tailored data extraction for different studies.

[9] The resulting list of tornadoes is available at https://minio.smtornado.com/minio/ui/browser/tables.

3.2. Approach 2

Using YOLO enables immediate retrieval of tornado coordinates and characteristics, in contrast to approach 1. Since this neural network does not take into account the rotation of a tornado, a more stringent selection may be required. Ultimately, the confidence threshold can be adjusted based on expert assessment to ensure more accurate identification. The PR curve for YOLO and the confusion matrix are shown in Figure 6. The standard mean average precision metric (T.-Y. Lin et al. 2014) is 0.524.

The importance of expert assessment can be illustrated using the example of a magnetic tornado observed on 2023 March 17, as shown in Figure 7. The YOLO model detects the tornado with a confidence score of 0.68; however, the corresponding bounding box is inaccurate, omitting the upper portion of the structure. Reviewing earlier frames reveals that the model detects a second bounding box on 2023 March 16 at 23:48, as illustrated in Figures 7(a) and (b). A similar issue arises while using the CRNN model. Rather than producing an incorrect bounding box, it may fail to detect the structure entirely within a given frame sequence. Additionally, because the CRNN operates on video sequences rather than individual images, it may label an entire sequence as containing a tornado, even when the structure is absent in some of the frames, particularly at the beginning or end of the sequence.

A visual inspection reveals the challenges faced by the model. For instance, the top part of the tornado appears with low contrast against the background, making it more difficult for both the model and the experts to recognize the structure. However, if several frames of a sequence are presented, the human brain perceives the video sequence and can correct the bounding box. As a result, to compile an accurate list of tornadoes, either putting less stringent restrictions on the confidence value or conducting an expert assessment by eye is required. For general purposes, we have set the probability cutoff at 0.5 for the CRNN model in the table and have kept the default detection parameters in Ultralytics.

In Figures 7(c) and (d), we provide an example of a correct detection with a high confidence score (0.74) on 2023 March 12 at 00:00 UTC. Although the portion of the tornado that is above the limb is rather small (around 12,000 km) and bent, the model still manages to detect it. Although many tornadoes lean toward the loops of the dominant magnetic field, this example shows that our system is capable of detecting the most "classic" stand-alone tornado structures. We evaluated the performance of the model and found that, although it may struggle with ambiguous cases or interrupted sequences, it reliably detects well-defined, isolated tornadoes. Therefore, YOLO models are suitable for the task of identifying solar

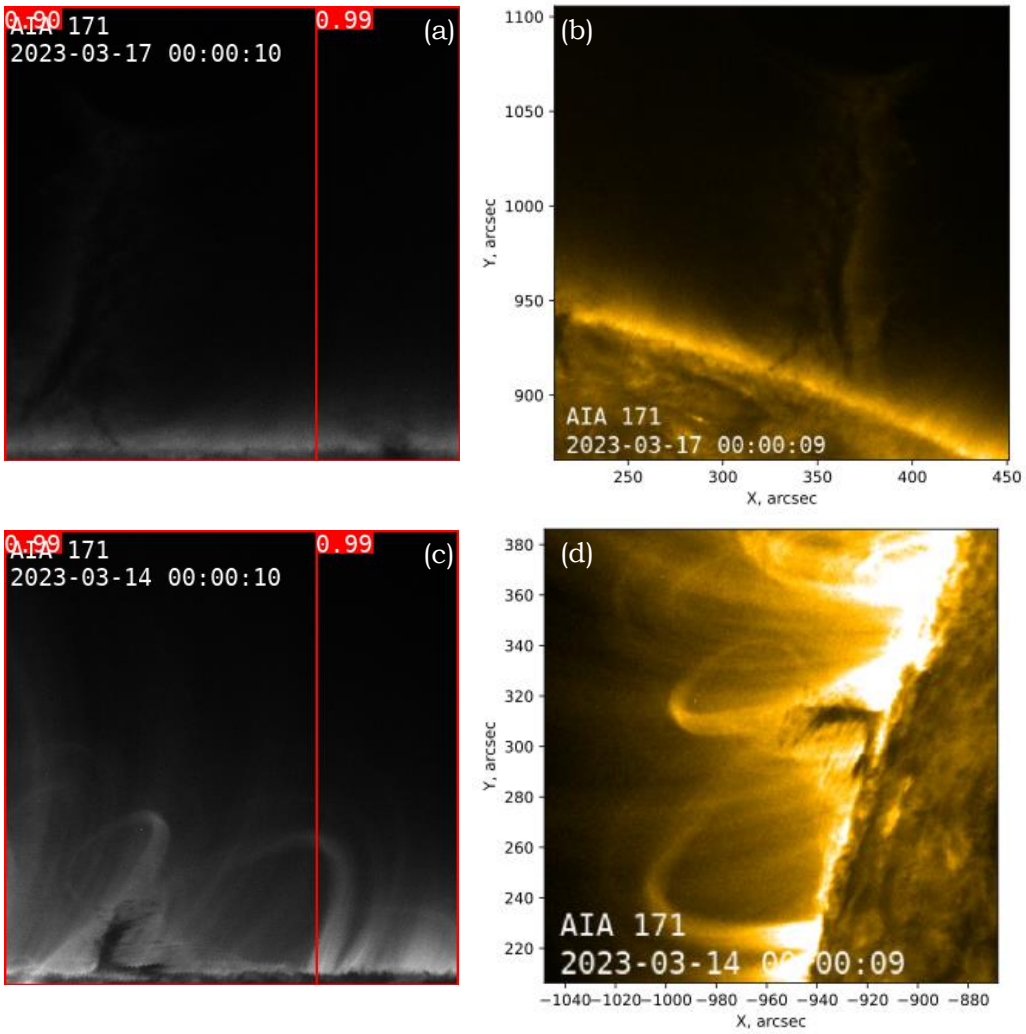


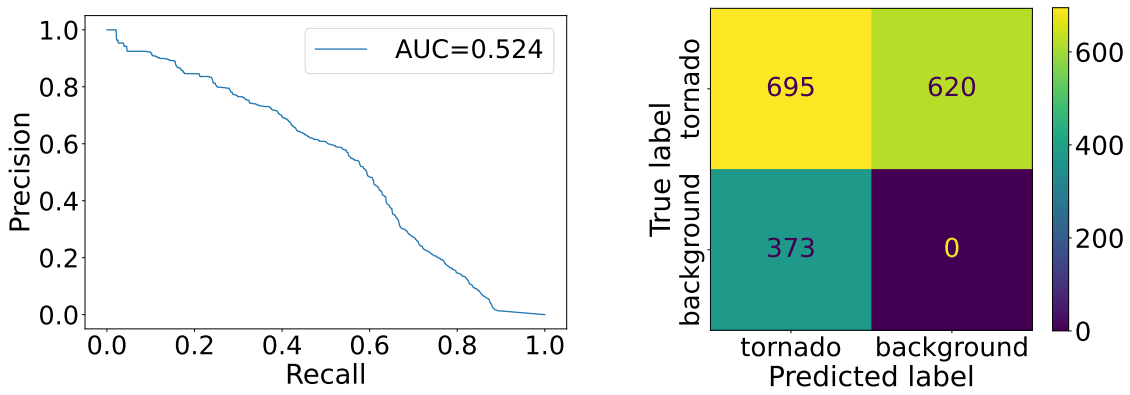
Figure 5. Examples of tornadoes identified by our model. (a) A large tornado detected on 2023 March 17, 00:00 UTC. Image with preprocessing for the CRNN model, with its confidence and overlap. (b) Colored version of (a) without any specific preparation for the neural network. (c) A tornado detected on 2023 March 14, 00:00 UTC, by the same model, with its confidence. (d) Analogous to (b). Wavelength, date, and time text on (a) and (c) was added separately and is not part of the preparation for the neural network. Animations of panels (b) and (d) are available at https://minio.smtornado.com/minio/ui/browser/gifs. The animations span 5 hr and 4 hr 54 minutes of observations, respectively, with a 6 minute cadence and illustrate the evolution and apparent rotation of tornadoes.
(An animation of this figure is available in the online article.)

magnetic tornadoes, particularly in applications requiring statistical analyses.

3.3. Statistical Analysis of Magnetic Tornadoes Identified by YOLO: Their Characteristics and Relation to Coronal Magnetic Topology

A separate topic for further research is the problem of nonrandom localization of magnetic tornadoes in the solar atmosphere. In particular, chains of magnetic tornadoes are known to be localized in the vicinity of neutral lines or so-called polarity inversion lines (PILs), as noted in S. Wedemeyer et al. (2013). In Figure 8, we show a bright example of the formation of the long-lived chain of magnetic tornadoes identified by both methods along the PIL associated with the filament of the quarter-Sun length. The upper panels show the development of the chain of tornadoes observed by SDO in the lower corona, and the lower panels provide the corresponding location of the filament associated with the tornadoes. The filament, representing a dense and cold chromospheric plasma structure, is recognized by the Automatic Solar Synoptic Analyzer (ASSA), which is an automated identification software system provided in the ASSA gallery.[10] Filaments are known to be closely related to PILs and always observed together (e.g., R. Mazumder et al. 2018). The eruption of the tornado chain seen in the video corresponds to the scenario proposed by I. Mghebrishvili et al. (2015) when the prominence tornado is located between two coronal loop systems at the evolving and rising PIL formed above the cavity. Magnetic reconnection occurring at the PIL leads to the eruption of the tornadoes as a part of the coronal mass ejection. This event illustrates the importance of identification of magnetic tornadoes in order to understand their link with active processes in the solar corona. In particular, compiling the list of numerous magnetic tornadoes opens an opportunity to analyze their statistical properties, location, and evolution with the solar cycle.

[10] https://iswa.gsfc.nasa.gov/IswaSystemWebApp/



(a) (b)

Figure 6. Key characterization curves for the YOLO model. (a) PR curve. (b) Confusion matrix.

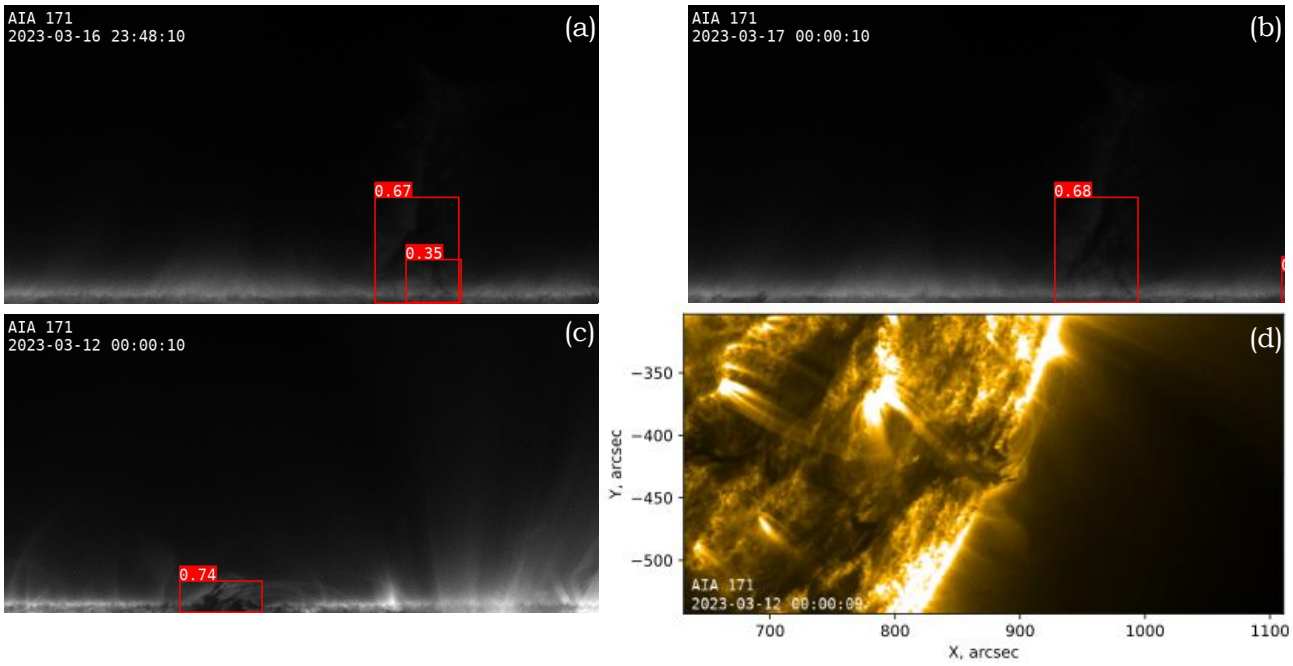


Figure 7. Example of identification of a large tornado on 2023 March 16–17 and associated confidences. (a) Two bounding boxes on 2023 March 16 at 23:48, with the larger one being incorrect. (b) One bounding box on 2023 March 17 at 00:00, still incorrect. (c) Example of the correct detection and bounding box on 2023 March 12 at 00:00 UTC. (d) Colored version of (c) without any specific preparation for the neural network. An animation of panel (d) is available at https://minio.smtornado.com/minio/ui/browser/gifs, spanning 4 hr and 54 minutes of observations with a 6 minute cadence.
(An animation of this figure is available in the online article.)

To demonstrate the scientific value of automatic detection techniques for solar magnetic tornadoes, we examine their spatial distributions during recent epochs of solar minimum and solar maximum, as identified by YOLO. We analyze the characteristics of the tornadoes for the Bartels rotation intervals no. 2536–2548 (minimum) and no. 2602–2614 (maximum), with each interval spanning approximately 1 yr (±6 months with respect to the solar minimum and maximum extrema in the sunspot numbers).

Downloading and processing the AIA data for a full year requires roughly 1 TB of storage, with the main bottleneck being the download speed. Once the data are available, running the neural network is comparatively fast, and our code supports parallel preprocessing across multiple CPU cores. The neural-network-based predictions for 1 yr of data can be completed within a few days.

The corresponding latitudinal distributions of magnetic tornadoes at the solar limb are shown in Figures 9(a) and (b), illustrating how their occurrence varies between different phases of the 11 yr solar cycle.

During the solar minimum, their distribution is notably symmetric and well structured, exhibiting four pronounced density maxima at approximately ±45° latitude (Figure 9(a)). In contrast, during the solar maximum, the distribution becomes highly irregular, characterized by numerous peaks with only slightly enhanced densities at lower latitudes (Figure 9(b)). A periodicity of approximately 15° remains discernible. To interpret these results, we compare the spatial distributions with typical coronal magnetic field configurations reconstructed via the PFSS model.[11] Figures 9(c) and (d) show

[11] https://sdowww.lmsal.com/sdomedia/SunInTime/webgl_tool/sv4/

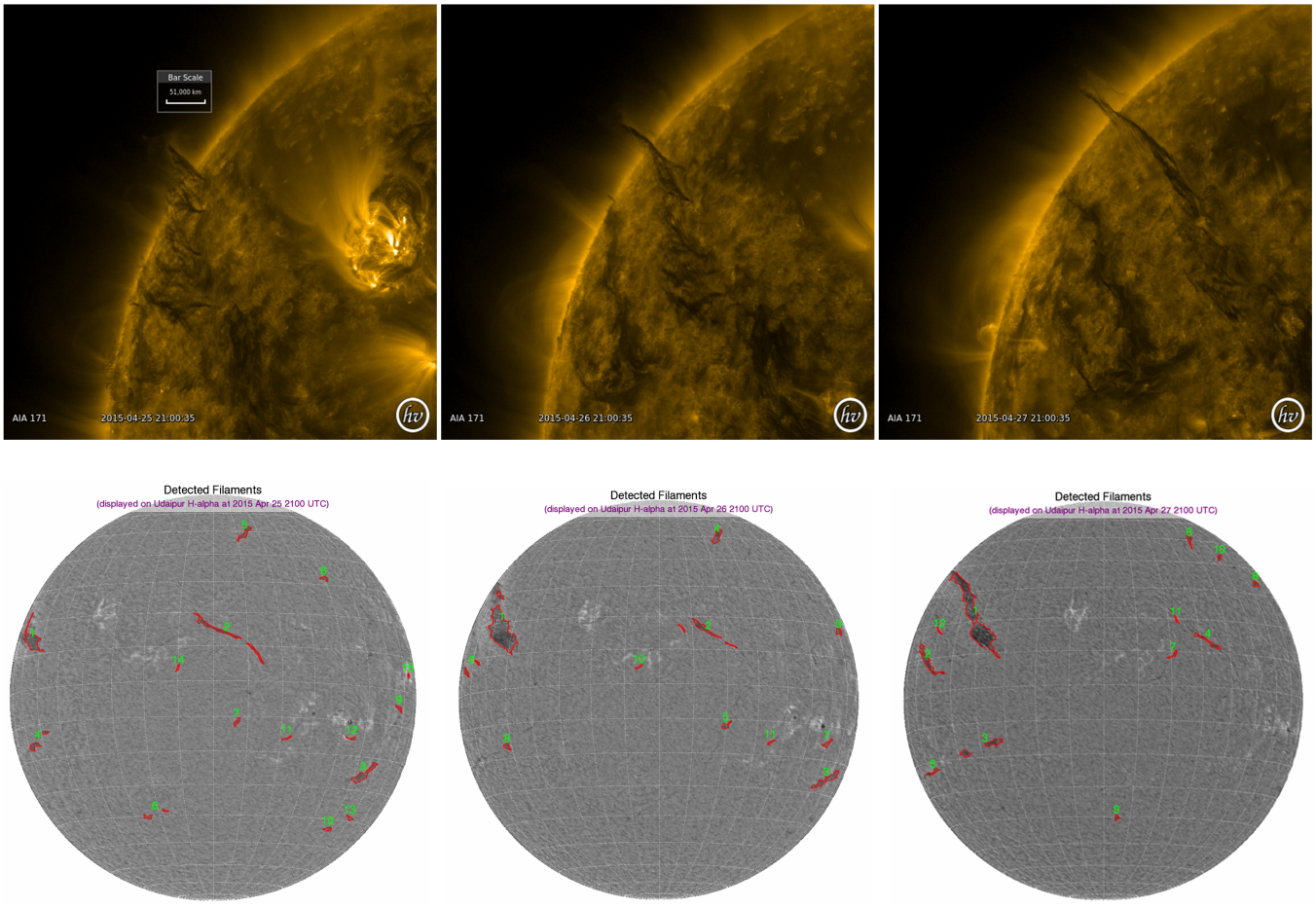


Figure 8. Example of formation of the chain of magnetic tornadoes at the PIL on 2015 April 25–27. Upper panels: three consequent days of observations of magnetic tornadoes. SDO AIA 171 Å data. Lower panels: the corresponding location of the largest filament observed in the solar disk (filament 1) associated with the magnetic tornado chain (identified by the ASSA model). The evolution of the chain of tornadoes is visible in the animation of the figure available at https://minio.smtornado.com/minio/ui/browser/gifs, spanning a week of observations.

(An animation of this figure is available in the online article.)

PFSS solutions for 2019 December 11 (minimum) and 2024 October 28 (maximum), corresponding to the central dates of the intervals used in panels (a) and (b). The purple and green colors denote opposite magnetic polarities, while white indicates closed magnetic field lines. Figure 9(c) illustrates the dipole-dominated configuration characteristic of solar minimum, with open magnetic field lines associated with polar coronal holes and closed field lines concentrated at low latitudes. The boundaries of high-latitude coronal holes lie near $\pm 45°$.

Comparing Figure 9(a) with Figure 9(c) reveals that the primary peaks in magnetic tornado density coincide with the edges of coronal holes. Their boundaries are known to be large-scale current sheets (O. Khabarova et al. 2021). Smaller peaks located within coronal holes, particularly in the northern hemisphere, likely correspond to polar conic current sheets that form during solar minimum within coronal holes (O. V. Khabarova et al. 2017). The existence of these conic current sheets depends on the stability of the parent coronal holes, and they disappear as coronal holes migrate equatorward during the rising phase of the solar cycle. Note that a perpendicular cut of the current sheet is a PIL, or, in other words, PILs are roots of current sheets.

Figure 9(d) illustrates the more complex and multipolar coronal magnetic structure characteristic of solar maximum. During this phase, coronal holes with open magnetic field lines coexist with active regions connected by closed coronal loops. Sunspots, visible as dark areas in Figure 9(d), mark the footpoints of strong magnetic fields in the photosphere and anchor many of the coronal loops. Neutral lines (PILs) form along the interfaces between regions of opposite magnetic polarity and frequently act as origins for the loop-forming field lines. Given the association between tornado density peaks and coronal hole boundaries during solar minimum, it is reasonable to hypothesize that the smaller-scale peaks in Figure 9(b) represent the locations between coronal loops where PILs are situated. These features exhibit a degree of periodicity of approximately 15° during solar maximum.

In summary, the spatial distribution of solar magnetic tornadoes indicates their preferential occurrence at (1) the boundaries of coronal holes hosting strong current sheets, (2) within coronal holes at the edges of polar conic current sheets, and (3) the footpoints of magnetic loops, which coincide topologically with neutral lines. These findings are consistent with earlier associations of large, long-lived magnetic tornadoes with PILs, and they highlight the need for further targeted investigation.

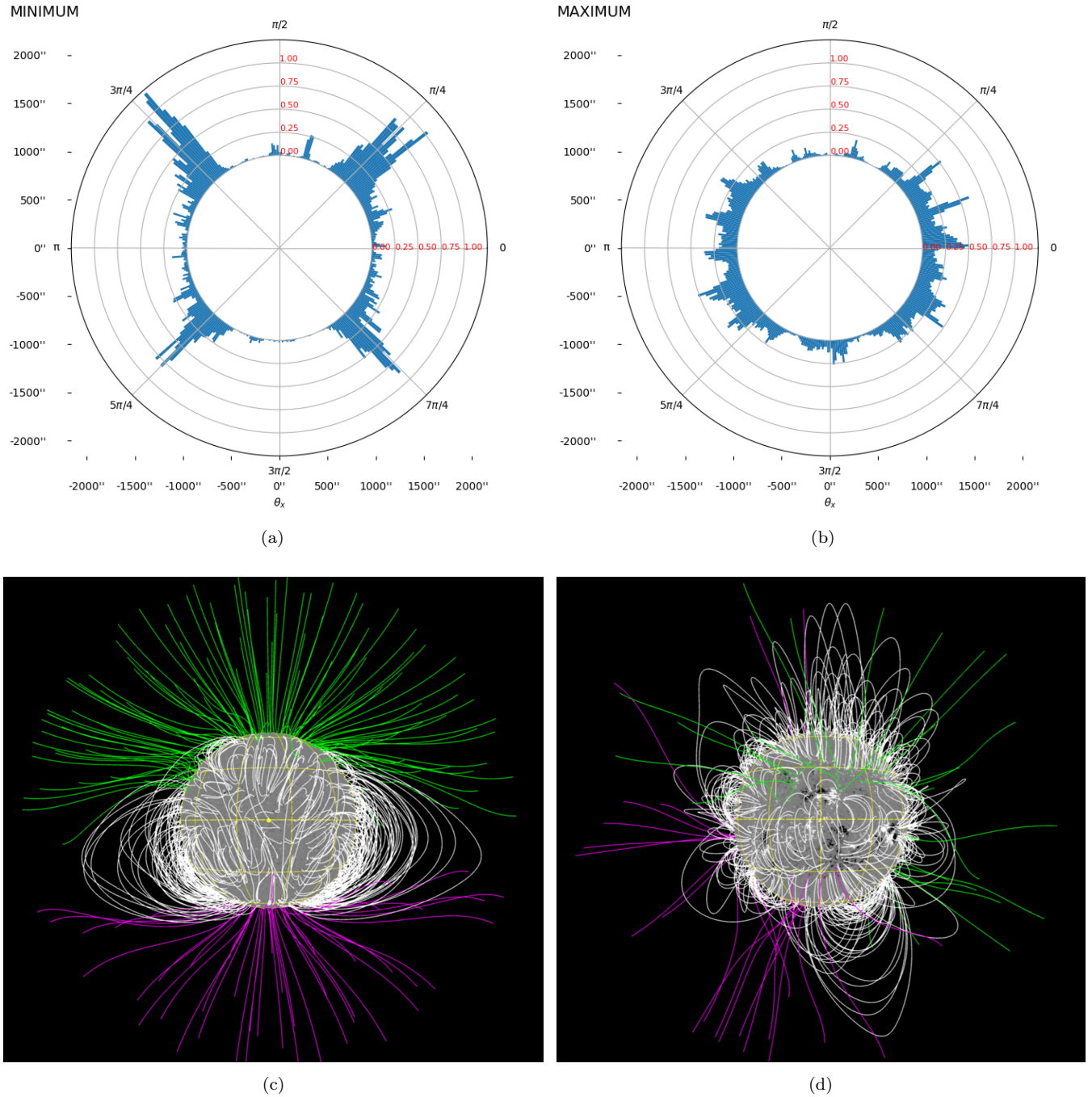


Figure 9. Spatial distribution of magnetic tornadoes in comparison with representative configurations of magnetic field lines in the solar corona during solar minimum and solar maximum. Panels (a) and (b) present the latitudinal distribution of magnetic tornadoes observed at the solar limb for Bartels rotation intervals no. 2536–2548 (solar minimum) and no. 2602–2614 (solar maximum), respectively. The histogram height corresponds to the density of events. Panels (c) and (d) show examples of coronal magnetic field line configurations reconstructed using the PFSS model for representative dates of solar minimum (2019 December 11) and solar maximum (2024 October 28). Magnetic tornadoes tend to cluster near the boundaries of polar coronal holes during solar minimum and between coronal loops during solar maximum, where PILs are typically formed.

4. Discussion and Conclusions

This work presents the development and application of an AI-based framework for the automated detection and characterization of solar magnetic tornadoes. Although tornado-like prominences have been observed for nearly a century and are of growing interest due to their potential role in energy transfer within the solar atmosphere, their underlying physics remains poorly understood. Another challenge is that these structures have traditionally been identified manually, which, while accurate, is a very time-consuming process, particularly when analyzing their temporal and spatial dynamics on a large scale, such as over a solar cycle. Furthermore, their spatial characteristics are complex and influenced by projection effects, oscillatory motions, and counter-streaming flows, all of which may contribute to misinterpretation.

As a result, reliable identification is generally limited to limb observations in the lower corona, which was the focus of our analysis.

The development of techniques for distinguishing solar magnetic tornadoes from the surrounding plasma structures in the solar corona is addressed for the first time. We employed two approaches: (i) a newly developed neural network combining convolutional, recurrent, and fully connected layers and (ii) the widely used YOLO neural network. The first method, which processes sequences of frames through a multiblock neural network, successfully identifies most tornadoes previously reported in the literature. Expert validation indicates an identification accuracy of approximately 85%. YOLO yields comparable results, performing well in detecting classic tornadoes but often failing when structures overlap. The CRNN and YOLO approaches employed in this study are complementary and address different aspects of solar magnetic tornado identification. The CRNN is particularly applicable to time-resolved image sequences, as it exploits both spatial features and temporal evolution, enabling the detection of rotating structures even when their morphology evolves gradually. However, it solves only a binary classification task and does not give exact coordinates for a tornado. In contrast, YOLO provides fast frame-based detection and performs reliably for well-defined, isolated tornadoes, but its lack of explicit temporal modeling limits its ability to identify complex or overlapping structures and capture rotational motion. Consequently, while both methods demonstrate strong potential, their applicability is constrained by different assumptions.

The integration of established tracking algorithms such as BoT-SORT (N. Aharon et al. 2022) or MIL (B. Babenko et al. 2009) may enhance tornado tracking and lifetime statistics. Our YOLO implementation does not account for rotation, aiming to demonstrate baseline model capabilities; however, combining YOLO with recurrent networks such as LSTM (S. Yun & S. Kim 2019) or ROLO (G. Ning et al. 2016) could improve performance. As recurrent layers tend to reduce processing speed (A. B. Qasim & A. Pettirsch 2020), any hybrid approach requires careful evaluation. Data management could also be improved by storing results in a database system like PostgreSQL, rather than separate CSV files. Furthermore, correlating outputs from both models, for example, confirming detections in overlapping regions, may increase confidence in tornado identification.

The results obtained in the study allow further analysis of the characteristics of solar magnetic tornadoes, which brings researchers closer to understanding the physical processes lying behind their formation and evolution in the solar corona.

The analysis of the spatial distribution of solar magnetic tornadoes during recent periods of solar minimum and solar maximum reveals a strong relationship between their occurrence and the global magnetic topology of the solar corona. Using our database, we could statistically confirm, for the first time, that solar magnetic tornadoes preferentially arise at sites marked by sharp gradients or topological transitions in the magnetic field, including coronal hole boundaries, polar conic current sheets, and loop footpoints associated with PILs. Such regions are well known to host magnetic reconnection leading to eruptive processes that release solar flares and where coronal mass ejections can occur. Therefore, further studying the spatial properties of magnetic tornadoes is of particular importance to understand the solar activity.

These results reinforce earlier observational links between persistent, large magnetic tornadoes and neutral line geometries, and they highlight the need for further coordinated studies that combine high-resolution observations, automated detection techniques, and advanced magnetic field modeling. A deeper understanding of the spatial behavior of magnetic tornadoes will provide valuable insight into the physical conditions driving reconnection and eruptive activity throughout the solar cycle.

The summary of our study can be formulated as follows.

1. Two neural network methods for identifying solar magnetic tornadoes have been developed, and two corresponding machine-learning-ready datasets have been created.
2. The first method uses a CRNN model that classifies videos, which allows taking into account the rotation of the structure.
3. The second method based on YOLO solves the detection problem and allows estimating the size of the structure.
4. A database of detected tornadoes has been created. There is an option to download limb images containing tornadoes via S3 storage.
5. Solar magnetic tornadoes may serve as tracers of reconnection-prone environments and potentially contribute to the energy transport or dynamics preceding eruptive events as they consistently occur at sites of strong magnetic gradients and topological transitions, including coronal hole edges, conic current sheets, and loop footpoints associated with PILs.
6. The spatial distribution of solar magnetic tornadoes differs between solar minimum and solar maximum. During solar minimum, density peaks coincide with the boundaries of coronal holes and smaller-scale polar conic current sheets. During solar maximum, the distribution is irregular, with peaks located between coronal loops where magnetic separators and PILs are typically found.

The proposed frameworks represent a step toward integrating AI tools into solar data analysis pipelines. Our results demonstrate that automated AI-based methods streamline the identification of solar magnetic tornadoes and uncover previously undetected structures, enabling comprehensive statistical studies and advancing our understanding of the role of these magnetoplasma structures in dynamic processes in the solar corona.

The neural-network-based framework developed in this study not only enables automated identification of solar magnetic tornadoes but also provides a foundation for broader applications in the analysis of dynamic solar limb phenomena. The two independent models, namely, a CRNN classifier that accounts for temporal evolution and rotational motion and a YOLO-based detector that localizes structures and estimates their size, capture spatiotemporal patterns that are characteristic of a wider class of magnetoplasma events, including chromospheric swirls, filament footpoint rotations, and vortex flows associated with small-scale eruptions.

This makes the models well suited for transfer learning, where knowledge gained from tornado detection can be adapted to related tasks with limited additional data and training. Combined with the machine-learning-ready datasets

and the publicly accessible archive of annotated limb images via S3 storage, this framework supports the development of more general-purpose tools for detecting and analyzing dynamic processes in the lower corona. Future extensions could include integrating additional observational channels, enabling multimodal analysis of the coupling between photospheric drivers and coronal responses. In this way, the methodology proposed here can contribute not only to the statistical study of solar magnetic tornadoes but also to a more comprehensive understanding of energy and mass transport in the solar atmosphere.

Acknowledgments

We thank the providers of observations obtained by the Solar Dynamics Observatory (SDO): http://sdo.gsfc.nasa.gov/. Filament location is identified with the ASSA software system provided by the Korean Space Weather Center of the National Radio Research Agency at https://iswa.gsfc.nasa.gov/IswaSystemWebApp/. We thank the Helioviewer website team (https://helioviewer.org/) for providing SDO images for a quick look and identification of the sizes of structures in the solar corona. The authors are grateful to Michael Fridman for his initial consultations on the selection of neural network models.

Code and Data Availability

All data from this study, including datasets and model weights, can be downloaded following the instructions provided in our repository. The code to replicate our results and use our models is available in our GitHub repository: https://github.com/markblumenau/solar-magnetic-tornado. Poetry is used for dependency management. Instructions for the basic usage of our models and download tools are included, as well as a demo Jupyter Notebook. Additionally, MiniO object storage is employed to store CSV files with detection tables. MiniO buckets allow anonymous access.

Software: JSOC (http://jsoc.stanford.edu/), PyTorch (A. Paszke et al. 2019), torchvision (TorchVision maintainers & contributors 2016), Astropy (Astropy Collaboration et al. 2013, 2018, 2022), SunPy (The SunPy Community et al. 2020), Numpy (C. R. Harris et al. 2020), OpenCV (G. Bradski 2000), PIL (F. Lundh & J. A. Clark 2026), Matplotlib (J. D. Hunter 2007; T. A. Caswell et al. 2023), Pandas (W. McKinney 2010; The pandas development team 2023), Ultralytics (G. Jocher et al. 2023), tqdm (C. da Costa-Luis et al. 2023), scikit-learn (F. Pedregosa et al. 2011), Jupyter-notebook (T. Kluyver et al. 2016), MiniO (https://www.min.io), Poetry (https://python-poetry.org).

Appendix A
Neural Network Components

A.1. Convolutional Layers

A convolutional layer (Y. Lecun et al. 1998) is a convolution operation, which is defined as follows: each $N \cdot N$ image fragment is multiplied element by element by a filter of the same size, the result is summed, a bias is added to it, and it is written to the corresponding position of the output matrix. Even with a single convolution, it is already possible to select objects that have a horizontal, vertical, or oblique dominant direction. Using nonlinear transformations and trainable convolutions, one can extract complex objects and details (M. D. Zeiler & R. Fergus 2014). Complex objects can be associated with a so-called feature vector (S. E. Umbaugh 2005).

Figure A1 shows the main block in the neural network used in the study. The block takes a tensor of size $h \times w \times k$ as input and returns a tensor of size $h/s \times w/s \times k'$, where s is a stride hyperparameter and k' is an output size hyperparameter. k and k' are the number of input and output channels, respectively. The block consists of three convolution layers: the first 1×1 convolution with an activation function (AF), the second DepthWise (A. G. Howard et al. 2017) convolution layer with an AF, and the third 1×1 convolution without any AF. There is a residual connection between the second and third layers, which is active only when $s \neq 1$. Using the depthwise convolution means that we apply a single convolutional filter for each input channel. After the depthwise convolution, a squeeze-and-excite (SE; J. Hu et al. 2018) connection is added. The SE connection starts with an average pooling layer that converts the input tensor into a vector equal in size to the number of channels of the output tensor after depthwise convolution by taking an average of all the pixels of the channel. Then there are two fully connected layers with the ReLU AF (K. Jarrett et al. 2009) and the hard-σ AF (K. Cho et al. 2014). A fully connected layer (D. E. Rumelhart et al. 1986) is defined in the following way: $y = xA^T + b$. Here, A is a weight matrix and b is a bias set by the optimization algorithm. The output of the last layer is effectively a vector of channel weights, indicating which channels are more important. Each element of the vector is multiplied by the

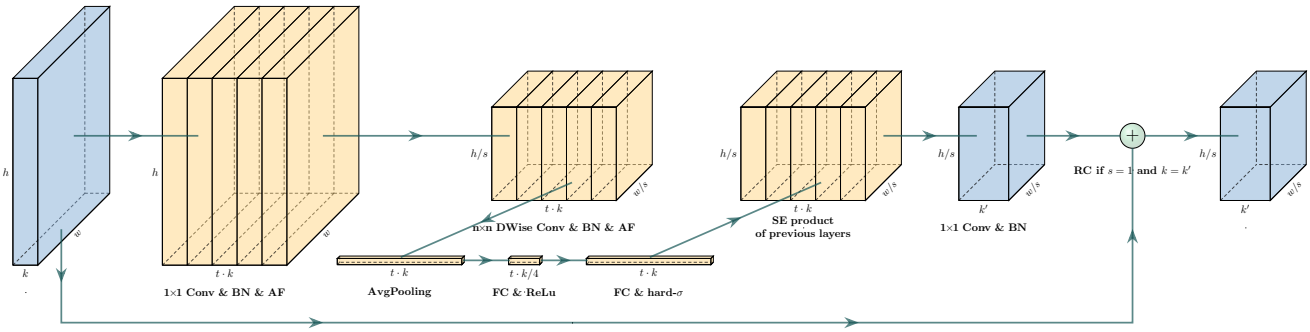
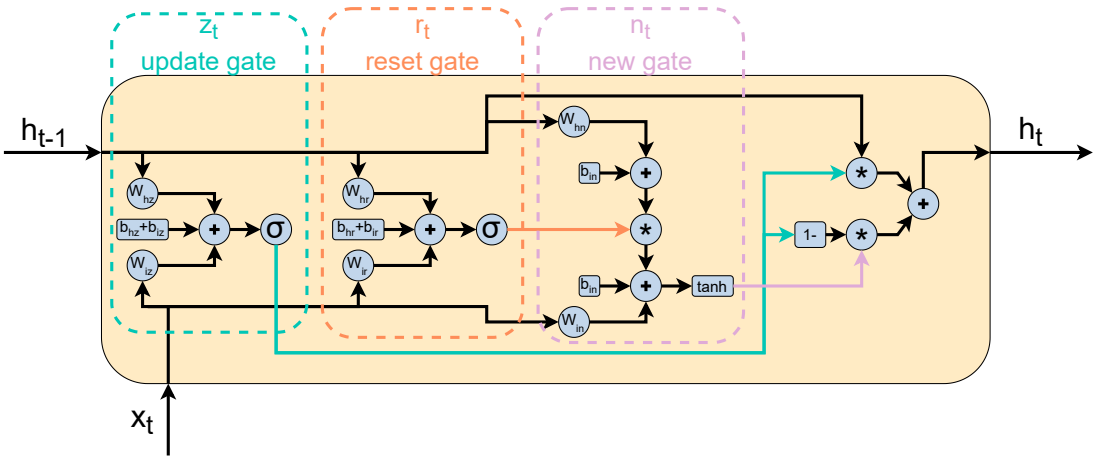
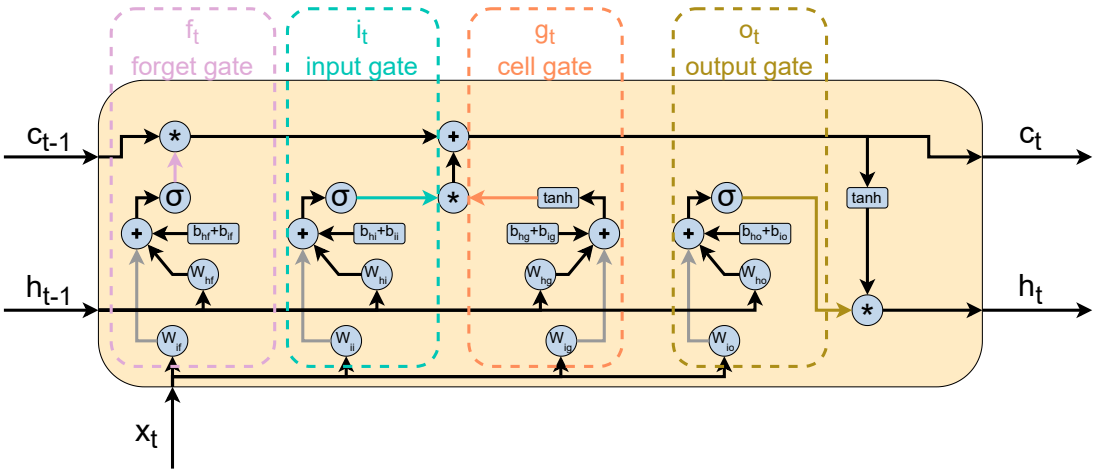


Figure A1. Schematic representation of the main block of the feature extractor CNN. The block receives a tensor of size $h \cdot w \times k$ as input, which is converted using a 1×1 convolution layer with Batch Normalization (BN) and AF, and then a DepthWise layer with an $n \times n$ size convolution gets converted into a tensor of size $h/s \times w/s \times k'$, where s is the stride. If $s \neq 1$, then the resulting tensor is converted using a 1×1 convolution layer; otherwise, it must be summed with the input tensor due to the residual connection after the 1×1 convolution layer. As a result, the output tensor has a size of $h/s \times w/s \times k'$. In addition to the previously mentioned operations, the block also has an SE connection. See A. Howard et al. (2019) for more details.



(a) GRU layer



(b) LSTM layer

Figure A2. A schematic diagram of GRU (a) and LSTM (b) layers. In both cases, h is the hidden vector, x is the input vector, different W are weights, different b are biases, tanh is the hyperbolic tangent, σ is the sigmoid function, and * is the Hadamar product. For the GRU layer (a), z is the update gate, r is the reset gate, and n is the new gate. For the LSTM layer (b), c is the cell gate, f is the forget gate, i is the input gate, g is the cell gate, and o is the output gate.

corresponding channel of the depthwise convolution output tensor. BatchNorm (BN) layers (S. Ioffe & C. Szegedy 2015), which recenter and rescale the data, are added between convolutional layers for faster and more stable training. We use PyTorch BN implementation in our work.[12]

A.2. Recurrent Layers

The GRU layer (K. Cho et al. 2014) is given by the following:

$$r_t = \sigma(W_{ir}x_t + b_{ir} + W_{hr}h_{(t-1)} + b_{hr}), \quad \text{(A1)}$$

$$z_t = \sigma(W_{iz}x_t + b_{iz} + W_{hz}h_{(t-1)} + b_{hz}), \quad \text{(A2)}$$

$$n_t = \tanh(W_{in}x_t + b_{in} + r_t * (W_{hn}h_{(t-1)} + b_{hn})), \quad \text{(A3)}$$

$$h_t = (1 - z_t) * n_t + z_t * h_{(t-1)}. \quad \text{(A4)}$$

Here, the different W are weight matrices, h_t is a hidden state at time t, r is the reset gate, z is the update gate, and n is a new gate. x_t represents an input at time t; in the case of several layers, the h_t of the previous layer is passed to the second and subsequent layers instead of x_t. The asterisk is the element-wise product (Hadamard product), $\sigma(x) = 1/(1 + e^{-x})$, and $\tanh(x)$ is the hyperbolic tangent.

Respectively, the LSTM layer (S. Hochreiter & J. Schmidhuber 1997) is given analogously:

$$i_t = \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{t-1} + b_{hi}), \quad \text{(A5)}$$

$$f_t = \sigma(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf}), \quad \text{(A6)}$$

$$g_t = \tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{t-1} + b_{hg}), \quad \text{(A7)}$$

$$o_t = \sigma(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho}), \quad \text{(A8)}$$

$$c_t = f_t * c_{t-1} + i_t * g_t, \quad \text{(A9)}$$

$$h_t = o_t * \tanh(c_t). \quad \text{(A10)}$$

Here, i is the input gate, f is the forget gate, c is the cell gate, and o is the output gate.

The provided formulae are for conceptual reference and completeness. Implementation-level differences exist between these formulations and their realization in practice in PyTorch, and all recurrent layers used in this work are implemented using the standard PyTorch GRU/LSTM modules, shown in Figure A2.

[12] https://docs.pytorch.org/docs/stable/generated/torch.nn.BatchNorm1d.html

A.3. Adam Algorithm

We use the Adam algorithm (D. P. Kingma & J. Ba 2014) for updating model weights:

$$\begin{aligned}
&\text{for} t = 1 \text{ to } \ldots \text{ do} \\
&\quad g_t \leftarrow \nabla_\theta f_t(\theta_{t-1}) \\
&\quad m_t \leftarrow \beta_1 m_{t-1} + (1-\beta_1) g_t \\
&\quad v_t \leftarrow \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \\
&\quad \hat{m}_t \leftarrow m_t / \left(1-\beta_1^t\right) \\
&\quad \hat{v}_t \leftarrow \frac{v_t}{\left(1-\beta_2^t\right)} \\
&\quad \theta_t \leftarrow \theta_{t-1} - \gamma \hat{m}_t / \left(\sqrt{\hat{v}_t} + \epsilon\right).
\end{aligned} \tag{A11}$$

Here, θ are the model weights, γ is the learning rate, β_1 and β_2 are the optimizer parameters, and m and v are the first and second moments, respectively.

Appendix B
Database of Newfound Tornadoes

We created an S3 storage, where we store the results from our model. There are two CSV tables with "_crnn" and "_yolo" in their names. For simplicity and the future possibility to improve, both tables have the same columns.

1. detection_id: a unique ID for a detection. it is generated using the UUID library in Python.
2. timestamp: date and time of the event.
3. artifact_s3: name of the initial limb picture.
4. angle: the angle is set from 0 to 2π, where 0 coincides with the Ox axis of Cartesian coordinates, increasing counterclockwise, similar to a unit circle overlaying an image of the Sun in Helioviewer. For the CRNN model, we use the center point of the $384 \cdot 384$ segment.
5. y_center: center of the bounding box on the y-axis. It is in meters from the center of the Sun. For the CRNN model, we use the center point of the $384 \cdot 384$ segment.
6. width: approximate width of the tornado in meters. For the CRNN model, it is the width of a $384 \cdot 384$ segment.
7. height: approximate height of the tornado in meters. For the CRNN model, it is the height of a $384 \cdot 384$ segment.
8. x_absolute: normalized angle value (the range is from 0 to 1).
9. y_absolute: normalized y_center value (the range is from 0 to 1).
10. width_absolute: normalized width value (the range is from 0 to 1 in terms of the whole limb picture).
11. height_absolute: normalized width value (the range is from 0 to 1 in terms of the whole limb picture).
12. yolo_confidence: YOLO model confidence in detection (range 0–1).
13. crnn_confidence: CRNN model confidence in detection (range 0–1).
14. verified_detection: a Boolean flag for a future possibility of human detection verification.
15. tornado_id: an ID of a tornado for a tracking algorithm implemented. The idea is to be able to group detections of one tornado together.
16. verified_tornado: a Boolean flag for a future possibility of human verification of a detection being a part of a specific tornado.
17. yolo_version: version of the YOLO model used.
18. crnn_version: version of the CRNN model used.
19. yolo_image_index: index of the $768 \cdot 384$ image on the limb.
20. crnn_image_index: index of the $384 \cdot 384$ image on the limb.
21. algorithm_version: version of the tornado tracking algorithm (for future use).
22. created: date and time the detection record was uploaded.
23. updated: date and time the detection record was updated. In case anything changes, this value will differ from "created."
24. metadata: any additional metadata in JSON format.

Some columns are simply conversions from others. We decided to keep and provide data almost up to the raw output of our models for ease of expert judgment.

ORCID iDs

Mark I. Blumenau https://orcid.org/0009-0001-9585-8565
Olga Khabarova https://orcid.org/0000-0002-3230-2033
Ilia S. Nikitin https://orcid.org/0009-0000-3915-0055
Dmitrii L. Vorobev https://orcid.org/0009-0002-2966-4241

References

Aharon, N., Orfaig, R., & Bobrovsky, B.-Z. 2022, arXiv:2206.14651
Al-Owais, A., Sharif, M. E., Ghali, S., et al. 2023, Neural Comput. Appl., 35, 15709
Alzubaidi, L., Zhang, J., Humaidi, A. J., et al. 2021, Journal of Big Data, 8, 53
Armstrong, J. A., & Fletcher, L. 2019, SoPh, 294, 80
Asensio Ramos, A., Cheung, M. C. M., Chifu, I., & Gafeira, R. 2023, LRSP, 20, 4
Astropy Collaboration, Price-Whelan, A. M., Lim, P. L., et al. 2022, ApJ, 935, 167
Astropy Collaboration, Price-Whelan, A. M., Sipőcz, B. M., et al. 2018, AJ, 156, 123
Astropy Collaboration, Robitaille, T. P., Tollerud, E. J., et al. 2013, A&A, 558, A33
Babenko, B., Yang, M.-H., & Belongie, S. 2009, in 2009 IEEE Conf. on Computer Vision and Pattern Recognition (IEEE), 983
Barczynski, K., Schmieder, B., Peat, A. W., et al. 2021, A&A, 653, A94
Bradski, G. 2000, CiteOpenCV, GitHub, https://github.com/opencv/opencv/wiki/CiteOpenCV
Caswell, T. A., Lee, A., de Andrade, E. S., et al. 2023, matplotlib/matplotlib: REL: v3.7.1, Zenodo, doi:10.5281/zenodo.7697899
Chen, H., Zhang, J., Ma, S., Yan, X., & Xue, J. 2017, ApJL, 841, L13
Cho, K., van Merrienboer, B., Bahdanau, D., & Bengio, Y. 2014, Proc. SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation (ACL)
da Costa-Luis, C., Larroque, S. K., Altendorf, K., et al. 2023, v4.66.1, Zenodo, doi:10.5281/zenodo.8233425
Fernandez Borda, R. A., Mininni, P. D., Mandrini, C. H., et al. 2002, SoPh, 206, 347
Grishin, K., Mei, S., & Ilić, S. 2023, A&A, 677, A101
Hamada, A., Asikainen, T., Virtanen, I., & Mursula, K. 2018, SoPh, 293, 71
Harris, C. R., Millman, K. J., van der Walt, S. J., et al. 2020, Natur, 585, 357
Hochreiter, S., & Schmidhuber, J. 1997, NComp, 9, 1735
Howard, A., Sandler, M., Chen, B., et al. 2019, in 2019 IEEE/CVF Int. Conf. on Computer Vision (ICCV) (IEEE), 1314
Howard, A. G., Zhu, M., Chen, B., et al. 2017, arXiv:1704.04861
Hu, J., Shen, L., & Sun, G. 2018, in 2018 IEEE/CVF Conf. on Computer Vision and Pattern Recognition (IEEE), 7132
Hunter, J. D. 2007, CSE, 9, 90
Illarionov, E., Kosovichev, A., & Tlatov, A. 2020, ApJ, 903, 115
Ioffe, S., & Szegedy, C. 2015, in PMLR. 37, Proc. 32nd Int. Conf. on Machine Learning, ed. F. Bach & D. Blei (PMLR), 448, https://proceedings.mlr.press/v37/ioffe15.html
Jarolim, R., Veronig, A. M., Hofmeister, S., et al. 2021, A&A, 652, A13
Jarrett, K., Kavukcuoglu, K., Ranzato, M. A., & LeCun, Y. 2009, in 2009 IEEE 12th Int. Conf. on Computer Vision (IEEE), 2146

Jocher, G., Chaurasia, A., & Qiu, J. 2023, ultralytics/ultralytics, v8.0.0, GitHub, https://github.com/ultralytics/ultralytics
Jozefowicz, R., Zaremba, W., & Sutskever, I. 2015, in Proc. 32nd Int. Conf. on Int. Conf. on Machine Learning—Vol. 37, ICML'15 (JMLR), 2342
Khabarova, O., Malandraki, O., Malova, H., et al. 2021, SSRv, 217, 38
Khabarova, O. V., Malova, H. V., Kislov, R. A., et al. 2017, ApJ, 836, 108
Kingma, D. P., & Ba, J. 2014, arXiv:1412.6980
Kluyver, T., Ragan-Kelley, B., Pérez, F., et al. 2016, in Positioning and Power in Academic Publishing: Players, Agents and Agendas, ed. F. Loizides & B. Schmidt (IOS Press), 87
Kuniyoshi, H., Shoda, M., Iijima, H., & Yokoyama, T. 2023, ApJ, 949, 8
Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. 1998, IEEEP, 86, 2278
Lemen, J. R., Title, A. M., Akin, D. J., et al. 2012, SoPh, 275, 17
Li, X., Morgan, H., Leonard, D., & Jeska, L. 2012, ApJL, 752, L22
Lim, B., & Zohren, S. 2021, RSPTA, 379, 20200209
Lin, T.-Y., Maire, M., Belongie, S., et al. 2014, arXiv:1405.0312
Linker, J. A., Heinemann, S. G., Temmer, M., et al. 2021, ApJ, 918, 21
Liu, J., Ji, C., Wang, Y., et al. 2024, ApJ, 972, 187
Luna, M., Moreno-Insertis, F., & Priest, E. 2015, ApJL, 808, L23
Lundh, F., & Clark, J. A. 2026, Pillow (PIL Fork), v12.2.0.dev0, PIL, https://app.readthedocs.org/projects/pillow/downloads/pdf/latest/
Mazumder, R., Bhowmik, P., & Nandy, D. 2018, ApJ, 868, 52
McKinney, W. 2010, in Data Structures for Statistical Computing in Python (SciPy)
Mghebrishvili, I., Zaqarashvili, T. V., Kukhianidze, V., et al. 2015, ApJ, 810, 89
Mghebrishvili, I., Zaqarashvili, T. V., Kukhianidze, V., et al. 2018, ApJ, 861, 112
Miller, A. 1993, VA, 36, 141
Mohsen, S. 2023, Multimed. Tools Appl., 82, 47733
Ning, G., Zhang, Z., Huang, C., et al. 2016, arXiv:1607.05781
Panasenco, O., Martin, S. F., & Velli, M. 2013, SoPh, 289, 603
Panesar, N. K., Innes, D. E., Tiwari, S. K., & Low, B. C. 2013, A&A, 549, A105
Paszke, A., Gross, S., Massa, F., et al. 2019, arXiv:1912.01703
Pedregosa, F., Varoquaux, G., Gramfort, A., et al. 2011, JMLR, 12, 2825
Pesnell, W. D., Thompson, B. J., & Chamberlin, P. C. 2012, SoPh, 275, 3
Pettit, E. 1943, ApJ, 98, 6
Powers, D. M. 2020, arXiv:2010.16061
Qasim, A. B., & Pettirsch, A. 2020, arXiv:2010.15740
Quan, L., Xu, L., Li, L., Wang, H., & Huang, X. 2021, Electronics, 10, 2284
Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. 2016, in 2016 IEEE Conf. on Computer Vision and Pattern Recognition (CVPR) (IEEE), 779
Rumelhart, D. E., Hinton, G. E., & Williams, R. J. 1986, Natur, 323, 533
Schmieder, B., Zapiór, M., Ariste, A. L., et al. 2017, A&A, 606, A30
Scholl, I. F., & Habbal, S. R. 2007, SoPh, 248, 425
Schrijver, C. J., & DeRosa, M. L. 2003, SoPh, 212, 165
Shannon, C. E. 1948, BSTJ, 27, 379
Sobel, I. 2014, Presentation at Stanford A.I. Project, https://www.researchgate.net/publication/239398674_An_Isotropic_3x3_Image_Gradient_Operator
Sokolova, M., & Lapalme, G. 2009, IPM, 45, 427
Su, Y., Gömöry, P., Veronig, A., et al. 2014, ApJL, 785, L2
Su, Y., Wang, T., Veronig, A., Temmer, M., & Gan, W. 2012, ApJL, 756, L41
The pandas development team 2023, pandas-dev/pandas: Pandas, v3.0.1, Zenodo, doi:10.5281/zenodo.7741580
The SunPy Community, Barnes, W. T., Bobra, M. G., et al. 2020, ApJ, 890, 68
TorchVision maintainerscontributors 2016, pytorch/vision, GitHub, https://github.com/pytorch/vision
Tziotziou, K., Scullion, E., Shelyag, S., et al. 2023, SSRv, 219, 1
Umbaugh, S. E. 2005, Computer Imaging (CRC Press)
Wan, Y., & Li, J. 2023, Complex Intell. Syst., 10, 2083
Wedemeyer, S., Scullion, E., Rouppe van der Voort, L., Bosnjak, A., & Antolin, P. 2013, ApJ, 774, 123
Wedemeyer, S., & Steiner, O. 2014, PASJ, 66, S10
Wedemeyer-Böhm, S., Scullion, E., Steiner, O., et al. 2012, Natur, 486, 505
Yun, S., & Kim, S. 2019, in 2019 19th Int. Conf. on Control, Automation and Systems (ICCAS), 94
Zeiler, M. D., & Fergus, R. 2014, in Computer Vision—ECCV 2014, ed. D. Fleet et al. (Springer), 818
Zhang, T., Hao, Q., & Chen, P. F. 2024, ApJS, 272, 5