A Benchmark Dataset for Tornado Detection and Prediction Using Full-Resolution Polarimetric Weather Radar Data

MARK S. VEILLETTE,[a] JAMES M. KURDZO,[a] PHILLIP M. STEPANIAN,[a] JOHN Y. N. CHO,[a] TONY REIS,[a] SIDDHARTH SAMSI,[a] JOSEPH MCDONALD,[a] AND NICHOLAS CHISLER[b]

[a] *Lincoln Laboratory, Massachusetts Institute of Technology, Lexington, Massachusetts*
[b] *Offutt Air Force Base, Omaha, Nebraska*

(Manuscript received 2 February 2024, in final form 8 November 2024, accepted 23 December 2024)

ABSTRACT: A weather radar is a primary tool used by forecasters to detect and warn for tornadoes in near–real time. To assist forecasters in warning the public, several algorithms have been developed to automatically detect tornadic signatures in weather radar observations. Recently, machine learning (ML) algorithms, which learn directly from large amounts of labeled data, have been shown to be highly effective for this purpose. Since tornadoes are extremely rare events within the corpus of all available radar observations, the selection and design of training datasets for ML applications are critical for the performance, robustness, and ultimate acceptance of ML algorithms. This study introduces a new benchmark dataset, Tornado Network (TorNet), to support the development of ML algorithms in tornado detection and prediction. TorNet contains full-resolution, polarimetric, level-II WSR-88D data sampled from 9 years of reported storm events. A number of ML baselines for tornado detection are developed and compared, including a deep learning (DL) architecture capable of processing raw radar imagery without the need for manual feature extraction required for traditional ML algorithms. Despite not benefiting from manual feature engineering or other preprocessing, the DL model shows increased detection performance compared to non-DL and operational baselines. The TorNet dataset, as well as the source code and model weights of the DL baseline trained in this work, is made freely available.

SIGNIFICANCE STATEMENT: Accurate and timely detection of tornadic signatures in radar data enables forecasters to issue timely warnings and preparedness measures, ultimately saving lives and reducing the devastating impact of tornadic storms. Machine learning (ML) has already been shown to be an effective tool for detecting key signals in radar that can be used to locate and track existing tornadoes. To further advance the state of the art in this area, this study presents a benchmark dataset that can be shared across the research community for the development and validation of ML-based algorithms for tornado detection. This work also provides a baseline deep learning (DL) model that is able to efficiently detect tornadic signatures in radar data.

KEYWORDS: Tornadoes; Radars/Radar observations; Deep learning; Machine learning

1. Introduction

The study of tornadoes and their detection/prediction is a well-explored area in mesoscale meteorology, with the focus being almost exclusively on meteorological radar. Multiple iterations of tornado detection algorithms have been implemented in the Weather Surveillance Radar-1988 Doppler (WSR-88D) Open Radar Product Generator (ORPG) over the years [Ryzhkov et al. 2005; Warning Decision Training Branch (WDTB) 2011; Brown and Wood 2012]. Although the current tornadic vortex signature (TVS) algorithm generally performs well from a detection standpoint, it tends to have fairly high false alarm rates, as shown later in this study. Radar-based tornado detection methods attempt to identify known tornadic signatures using rule-based algorithms and, in some cases, have been used as training tools for forecasters. For example, although not currently operational in the ORPG, Ryzhkov et al. (2005) define the tornadic debris signature (TDS) using reflectivity factor, radial velocity, and polarimetric radar data. The TDS can be used to confirm a tornado is ongoing using WSR-88D data in near–real time. However, in many cases, tornadic debris is not lofted into the radar resolution volume due to weak tornado strength, lack of debris source, or overshooting of the radar beam due to range via the curvature of Earth and the atmospheric refractive index (Doviak and Zrnić 1993).

Researchers have turned to artificial intelligence (AI) and machine learning (ML) methods in recent years in an attempt to mitigate these issues and better detect tornadoes, especially when traditional algorithms have either failed or had too many false alarms (e.g., Lagerquist et al. 2020). Cintineo et al. (2018 and 2020) combined numerical weather prediction (NWP) models, satellite, radar, and lightning data to forecast severe weather, first in an empirical way and then followed up in a Bayesian classifier framework. Their algorithm, ProbSevere v2.0, is now operational in the Multi-Radar Multi-Sensor (MRMS) system (Smith et al. 2016). More recently, Sandmæl et al. (2023) and Zeng et al. (2022) utilized random forests to perform tornado detection in S-band radar. A series of features were designed that extracted statistical values from radar data, and these features were used to train the random forest model to determine the likelihood of a tornado being present. In each of these cases, as with many AI/ML

Corresponding author: Mark S. Veillette, mark.veillette@ll.mit.edu

DOI: 10.1175/AIES-D-24-0006.1 e240006

algorithms in meteorology, the results are increasingly impressive; however, the datasets and models are not always easily accessible to the broader community.

Within the fields of AI/ML, a significant proportion of the effort to develop algorithms and models lies within the process of dataset curation (Bishop 2007). Building datasets is a key task that often leads to the success (or failure) of an AI/ML approach. Within the AI/ML fields, the concept of benchmark datasets has become increasingly common in order to partially deal with the difficulties that lie within the data curation phase of research (e.g., Olson et al. 2017; Wu et al. 2017; Hu et al. 2020). Benchmark datasets are carefully curated, public data repositories that are shared with the world in order to accelerate the pace of research and development (Thiyagalingam et al. 2022). Instead of spending significant time developing a dataset, researchers are able to start at a baseline to build what they need, whether that involves augmenting the dataset or developing novel techniques to work with the existing dataset. Another key benefit of benchmark datasets is that research teams around the world can benchmark their models and results on the same baseline for fair comparisons. These datasets are often used as "challenge problems," where research teams are challenged to beat the current best-performing models with a given benchmark dataset (Dueben et al. 2022; Gadepally et al. 2022). Given the goal of making such datasets well organized and easily shareable/reusable, it is generally beneficial to adhere to the findable, accessible, interoperable, and reusable (FAIR) principles, which are guidelines for making data findable, accessible, interoperable, and reusable (Wilkinson et al. 2016). In doing so, benchmark datasets are capable of becoming an organic starting point for researchers looking to integrate data into new concepts and methodologies.

Within the field of meteorology, benchmark datasets are rapidly growing in popularity. Dueben et al. (2022), who specifically call out the need for benchmark datasets in the atmospheric sciences, acknowledge the incredible size of Earth science datasets and the common lack of direct applicability to existing benchmark image datasets like ImageNet (Russakovsky et al. 2015). Dueben et al. (2022) define a benchmark dataset as either "scientific" or "competition" types, with the former being more in line with research problems such as what has been described above and the latter being more about the broader community and nondomain experts. However, in the authors' experience, some datasets can serve as both (e.g., Veillette et al. 2020), allowing nondomain experts to contribute innovative new approaches. Dueben et al. (2022) also define order-1 and order-2 scientific benchmarks, with order-2 defining a "complete" benchmark. The list of benchmark datasets in meteorology continues to grow (e.g., Rasp et al. 2020, 2023; Betancourt et al. 2021; Haupt et al. 2021; Prabhat et al. 2021), and recent publications have summarized approaches to algorithm development that could benefit from benchmark datasets (e.g., Chase et al. 2022, 2023).

This study introduces a benchmark dataset, called Tornado Network (TorNet), for the development of ML algorithms for tornado detection and prediction. TorNet includes full-resolution, polarimetric, level-II WSR-88D data subsampled from 9 years of storm reports. The data are publicly available and meet most if not all of the Dueben et al. (2022) requirements for an order-2 benchmark dataset in Earth sciences. The availability of full-resolution imagery provides flexibility to explore and compare a range of different ML (and non-ML) algorithms. Of particular interest are deep learning (DL) algorithms, e.g., convolutional neural networks (CNNs), which can automatically detect low-level features from high-resolution imagery without the need for manual feature engineering. Construction and training of DL algorithms are typically more complex compared to non-DL algorithms like random forests, and thus, the availability of a benchmark dataset will allow researchers to more rapidly advance state of the art in this space. As a demonstration, a CNN baseline for tornado detection trained using TorNet is provided in this work, and this baseline is shown to outperform other common non-DL algorithms.

The paper is organized as follows. Section 2 provides the rationale, sampling procedure, and processing pipeline for creating TorNet. ML applications of the dataset are explored in section 3, where several ML baselines for tornado detection are developed and compared. Finally, the application of the CNN baseline model on selected case studies is provided in section 4.

2. Dataset description

This section provides the rationale and methodologies for creating TorNet, a benchmark dataset for the study of tornadoes containing full-resolution polarimetric radar data. This dataset encapsulates a broad variety of convective modes and storm types, including active confirmed tornadic storms, pretornadic storm evolution, nontornadic rotating storms, nonrotating severe storms, and nonsevere storms. This dataset was constructed to enable two primary research efforts: 1) supporting analysis and algorithm development for tornado detection by providing labeled examples of tornadic, rotating nontornadic, and nonrotating storms; 2) including time evolution of storms across the continuum of rotation intensities to support tornado prediction by providing potential precursors to tornadogenesis.

The following subsections provide details on the structure of the dataset, methodologies for selecting samples, and the data curation process. Links for downloading TorNet as well as code samples for working with the data are provided in the data availability section.

a. *Dataset structure*

TorNet is comprised of a large number of samples, which are the basic units of the dataset. Each sample contains a small cropped section, or chip, of six WSR-88D variables centered on selected locations (x_i, y_i) and times t_i (described below), where i represents the sample index. Each variable is provided as an array with axes of range, azimuth, time, and sweep. All samples in TorNet are selected from storm events found in the National Centers for Environmental Information's (NCEI's) Storm Events Database (NSED) (NCEI 2024). Time stamps, or frames, within each sample are assigned a binary label of either "tornadic" or "non-tornadic"

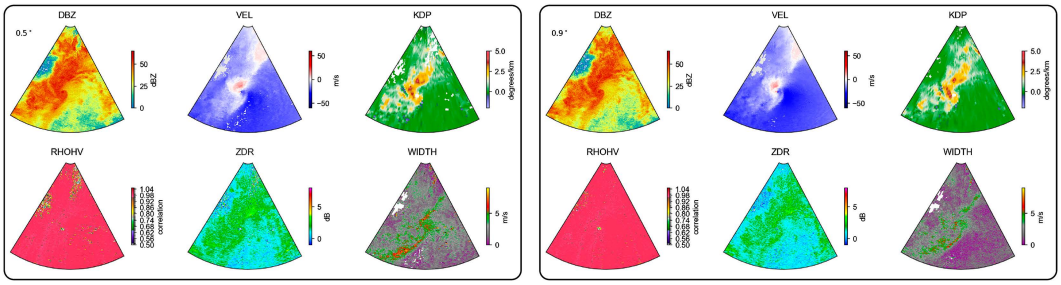


FIG. 1. One sample chip from the TorNet dataset. Each sample contains imagery of two elevation sweeps (0.5° left and 0.9° right) containing six radar variables: reflectivity factor (DBZ), radial velocity (VEL), specific differential phase (KDP), correlation coefficient (RHOHV), differential reflectivity (ZDR), and spectrum width (WIDTH). Each sample contains radar variables provided on a 60° by 60-km region (in azimuth and range, respectively) centered over sampled locations and times. The sample shown depicts one time frame of an EF3 tornado near KGWX, Tennessee, United States, on 23 Feb 2019 (NSED event ID 799239).

based on whether a confirmed tornado was present at the corresponding time stamp. Within NSED, storm events are also grouped into storm episodes. In most cases, multiple TorNet samples are drawn from a given storm event, and similarly, multiple storm events may be selected from a given storm episode. While each sample corresponds to a unique chip location and time, it is not uncommon for samples in TorNet to have some overlap in time and/or space.

Figure 1 provides an example of one frame from a TorNet sample. In this visualization, the data are plotted in Cartesian coordinates; however, it is important to note that the underlying data are represented in polar coordinates (range, azimuth). The rationale for not resampling data to Cartesian in TorNet was to preserve the native resolution of the data, especially when observations are close to the radar.

b. *Sample categories*

In general, tornadoes are extremely rare events. This leads to a class imbalance that is associated with well-known challenges in algorithm design (Johnson and Khoshgoftaar 2019; Ali et al. 2019; Tsagalidis and Evangelidis 2022). To serve as a useful benchmark for classification and prediction tasks, TorNet attempts to include as many tornadic samples as possible, while also maintaining a realistic imbalance between tornadic and nontornadic categories. This ensures that false-positive rates can be adequately assessed and that different strategies for addressing class imbalance (e.g., oversampling the minority class) can be considered. Moreover, the nontornadic cases in the dataset should contain a sufficient number of "difficult" cases, i.e., nontornadic cases that exhibit certain features that might lead to false warnings, such as a rotation signature in the velocity field. Such cases are also quite rare (in general), and not accounting for these may lead to algorithms that overdetect tornadoes.

To achieve a suitable balance of tornadic and nontornadic cases, samples in TorNet were selected based on three categories: confirmed tornado, tornado warning, or nontornadic random cell. The definitions of these categories (referred to as "confirmed," "warnings," and "random") and their rationale are as follows:

- *Confirmed tornado*: These are a selection of storm events from NSED with tornadoes confirmed by storm surveys. These events are gold-standard tornado observations and are useful for identifying and delineating the radar signatures indicative of existing tornadoes or tornadogenesis. Samples in this category receive a label of tornadic.
- *Tornado warning*: These are events where tornado warnings were issued by an NWS forecaster, but no tornado was confirmed. Samples from this category were added so the dataset contains samples that have high tornado potential (in terms of features in the radar data), but never were confirmed to produce a tornado. These storms are useful for identifying and delineating the radar signatures that are indicative of failed tornadogenesis. For the purposes of training ML models and computing benchmark scores, samples in this category are given a label of non-tornadic.
- *Nontornadic random cell*: These are a wide selection of all other nontornadic precipitation cells, ranging from pop-up convective showers to stratiform rain, nonrotating severe thunderstorms, mesoscale convective systems without embedded rotation, etc. These storms are useful for characterizing the broad range of radar signatures associated with precipitation and delineating these signatures from rotating—and potentially tornadic—storms. Samples in this category also receive a label of non-tornadic.

While a label of nontornadic is given to frames in the warning and random category, it is possible that some null samples in TorNet correspond to an actual tornado, as a nontrivial number of tornadoes occur but go unreported (Potvin et al. 2022). A strategy for dealing with possibly mislabeled samples is utilized for one of the ML baselines presented in section 3, and further discussion of this issue is provided in section 5.

c. *Event selection*

This section provides details on how events were selected from each of the categories presented in the previous section. For each category, the output of this phase is a list that contains sample locations (x_i, y_i, t_i) that contain 1) spatial locations x_i, y_i for each sample chosen to be either near the approximate location of a confirmed tornado (as described

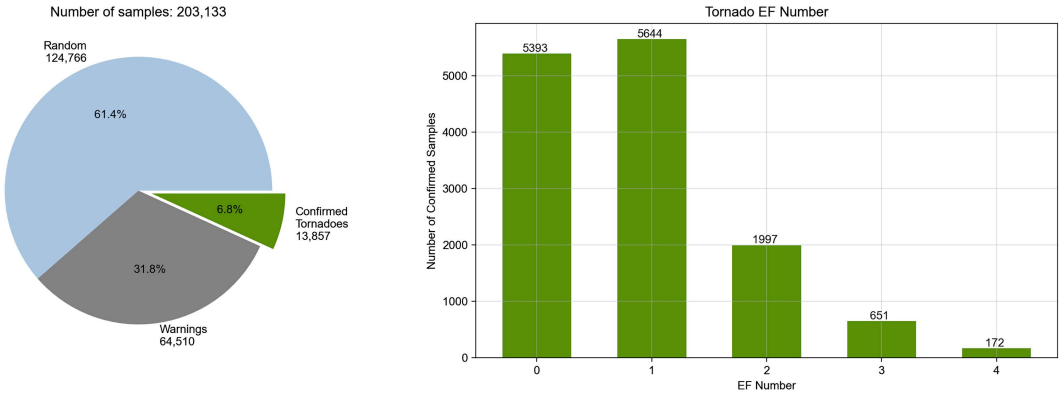


FIG. 2. (left) Numbers of samples that were created for categories of confirmed tornadoes, nontornadic tornado warnings, and nontornadic random cells. (right) Counts of samples for each EF number. Note that no EF5 tornado events occurred during the period spanning August 2013–22.

below) or in the vicinity of a tornado warning/severe storm report; and 2) time stamps t_i of each sample chosen at points within an event. All sample times in TorNet were from the period of August 2013–22.

For the confirmed category, the NSED formed the basis of finding tornadic events. For this category, a crude approximation of the location of the tornado was formed by linearly interpolating between the start and end points at 1-min temporal resolution. Points (x, y, t) along this path were recorded for the duration of the tornado. For the warning category, events were chosen using tornado warning polygons obtained from the Iowa State University warning archive[1] and removing any warnings that coincided with confirmed tornado observations within 30 min. Warning polygons were reduced into linear "tracks" (x, y, t) by drawing a line through the polygon centroid at the mean climatological azimuthal angle (41.2°) of tornadic tracks, which was computed from all contiguous U.S. tornadoes over 1998–2020. The start and end points of these warning tracks were defined by the western and eastern intersections of the line with the polygon and by the start and end times of warning validity. Sampling along these simulated tracks was done to provide more variety in TorNet samples and to increase the likelihood that resulting samples contain tornadic features.

For the random category, two methods were used for finding sample locations. First, severe thunderstorm warning polygons were obtained and filtered to exclude those that were collocated with tornado warnings and/or confirmed tornadoes. The center point of the warning, along with the issue time of the warning, was used as the sample location. Random locations and times from warning events that were far from warning polygons were also included in this category.

These selection procedures generate a large list of events from the confirmed, warnings, and random categories that form the basis of the TorNet dataset. At this stage, the number of warnings and random cases far outnumber confirmed cases. To reduce this to a final list of sample locations, the list was filtered to prevent cross contamination among categories and to create a balanced set of categories. The following considerations were made when sampling storm events across the three main categories:

1) *Confirmed*: These are the rarest storms, so we keep all available events.
2) *Warning*: These cases are filtered (±30 min) to ensure they do not overlap with any confirmed tornado cases, such that any random storm cell selected throughout the event will not be a confirmed tornado (but may potentially be another co-occurring warned but nontornadic cell).
3) *Random*: These cases are filtered (±30 min) to not overlap with any confirmed or warning events, such that any random storm cell selected throughout the event will be less likely to have tornadic characteristics (i.e., was not associated with any tornado warning, whether confirmed or false). These events are the most common, so we subset the resulting storms to achieve a balanced dataset.

After performing these steps, the final counts of samples drawn from each category are shown in Fig. 2. The final dataset contains 203 133 samples, with approximately 6.8% being from confirmed tornadoes. Of the nontornadic cases, approximately one-third are sampled from the warning category. Figure 2 also breaks down the confirmed tornado category by the EF number of the associated tornado track, showing that EF0 and EF1 tornadoes dominate the confirmed tornado datasets, with decreasing counts for EF2, EF3, and EF4.

d. *Radar image processing*

Given the final list of sample locations, corresponding observations are gathered from the nearest WSR-88D site. The variables extracted for each event include reflectivity factor (DBZ), radial velocity (VEL), spectrum width (WIDTH), differential reflectivity (ZDR), correlation coefficient (RHOHV), specific differential phase (KDP), and the storm cell identification and tracking (SCIT) information (Johnson et al. 1998), which is made up of a storm cell's location and unique identifier code. Radar tilts corresponding to 0.5° and 0.9° elevations for DBZ,

[1] https://mesonet.agron.iastate.edu/vtec/search.php.

VEL, WIDTH, RHOHV, and ZDR were extracted from WSR-88D level-II files at 0.5° azimuthal resolution and downloaded from Amazon Web Services (AWS) Open Data Registry. The SCIT identifiers and KDP fields were obtained from WSR-88D level-III archives obtained from Google Cloud Storage. Descriptions of these variables are provided in appendix C (see Fig. C1).

After downloading the required datasets for selected events, various preprocessing steps were performed in order to clean, align, and down-select the data. Given time t, the nearest available data for each variable were located for time t, $t-5$, $t-10$, and $t-15$ min. For each time, the level-II radial velocity field is dealiased using the method described in Veillette et al. (2023). KDP, which is provided at 1° azimuthal resolution in level-III archives, is upsampled to 0.5° using nearest-neighbor interpolation. All fields are aligned across range and azimuth dimensions. At this stage, all variables are available on axes (time, sweep, range, azimuth).

The final processing step involves subsampling from the range and azimuth dimensions so that dataset samples focus on smaller image chips centered over the locations (x, y) chosen in section 2c. This subsampling greatly reduces the size of the dataset and retains the most meaningful portions of the imagery. To create a sequence of image chips, the SCIT object closest to the chosen location (x, y, t) is selected. Radar variables corresponding to the selected SCIT are obtained, and these data are cropped in the (range, azimuth) dimensions to create a smaller image centered over the selected SCIT. The size of the chips used in TorNet was selected to be 60 km (240 250-m gates) by 60° (120 0.5° azimuthal intervals) by two sweeps containing 0.5° and 0.9° tilts. For a given look-up time t, the radar sweeps with the nearest time prior to t were selected, accounting for split-cuts and Scanning Doppler Radar with an Integrated Low-Power Auxiliary for Super-Resolution (SAILS) cuts [Office of the Federal Coordinator for Meteorology (OFCM) 2021]. The spatial dimensions were chosen to be large enough to capture tornado locations that do not closely align with SCIT locations (which is expected to happen in larger storms, especially supercells) or for highly nonlinear tornado paths. The same SCIT location is used for each time step in the sample.

After downsampling, the radar variables in each sample can be represented as a four-dimensional array with shape (T, S, R, A), where $T = 4$ corresponds to the number of time steps, $A = 120$ and $R = 240$ are the azimuthal and radial sizes of the chip, and $S = 2$ corresponds to the number of radar sweeps (or tilts) contained in the sample. Each sample is also assigned a label of size T denoting whether a confirmed tornado existed within 3 min of each time step. Some samples in TorNet have a mix of tornadic and nontornadic frames, e.g., a case where the tornado forms in the last frame of the sample. Each sample is saved to a separate netCDF file that includes the six radar variables, labels, and associated coordinate values. These files include a variety of metadata, including the sampling category, range-folded masks, event and episode identifiers (IDs) from the NSED, the EF number of confirmed tornadoes, and information about the NEXRAD site from which data were extracted.

3. Machine learning applications

TorNet was designed for ML applications in tornado detection and prediction. ML tasks that can be studied using the dataset include classification (tornado detection), time-series prediction (forecasting), explainable AI (XAI) methods, automated feature extraction, variable importance, and unsupervised learning. Moreover, TorNet contains all the necessary metadata required for augmenting the radar data with other observations and/or NWP forecasts.

To demonstrate the utility of TorNet, the remainder of this section will provide a number of baseline models for tornado detection. In this task, the final frame within TorNet samples is classified as tornadic or nontornadic. We begin by describing how TorNet is split into training and testing partitions and then describe and compare various classification baselines.

a. *Partitioning data into training and testing*

To properly assess hold-out detection performance, TorNet must be split into training and testing partitions. Any splitting approach needs to avoid leakage, which is when testing samples are similar/highly correlated with certain training samples. For this reason, a naive random splitting approach is not recommended, as samples often overlap in time and/or space. A more acceptable methodology for splitting the dataset is by assigning certain intervals of time to training and other time intervals to testing. This approach requires choosing optimal lengths of time for performing this partition. Intervals that are too small risk leakage from occurring at the edges of intervals, whereas intervals that are too long may leave out important seasonal variations from training and may lead to poorer generalizability and/or less reliable test results.

The split used in the following attempts to maximize seasonal variability in both training and testing, while minimizing data leakage. Each sample in the dataset is first associated with the start time t_e of its storm episode ID found in the NSED. Generally, storm episodes consist of groupings of storm events in a given location and time interval, so that events (and hence individual samples from those events) assigned to different episodes should be sufficiently separated in both time and location. If $J(t)$ represents the Julian day of the time stamp t, defined as the number of days since 1 January of the calendar year, samples with $J(t_e) \pmod{20} < 17$ were considered training data, and samples with $J(t_e) \pmod{20} \geq 17$ are considered test samples. As a final precaution against cases where storm episodes overlap, any training sample that was within 30 min and within 0.25° of latitude/longitude of any test sample was removed from TorNet. This method allocated 171 666 (84.5%) samples into training and 31 467 (15.5%) samples into testing. This method of splitting ensures that both training and testing provide coverage for the entire 10-yr span of TorNet.

b. *Baseline models*

The baseline models include three ML algorithms (logistic regression, random forest, and CNN) and a non-ML algorithm (tornado vortex signature). Two of the ML models, namely, logistic regression and random forest, require a feature

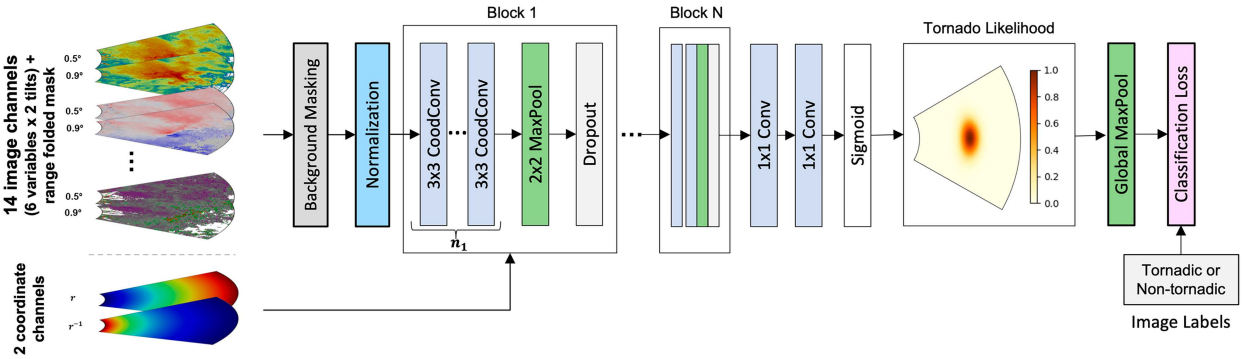


FIG. 3. DL architecture used for the tornado detection. Multiple radar modalities shown on the top left are stacked along the channel dimension. Background values are flagged, and channels are normalized to the range [0–1]. The normalized data are processed by multiple blocks that utilize CoordConv convolution layers which also ingest the radial coordinates of the images. The network produces a map of tornado likelihoods, which is processed with a global maximum pooling layer before computing the loss function.

extraction step where new features are estimated (e.g., computation of the azimuthal shear field), and full-resolution imagery associated with each sample is reduced to a feature vector of fixed length. This study implemented a feature extraction approach similar to that described in Sandmæl et al. (2023), with some modifications related to input data sources, processing steps, and feature definition. The details of the feature extraction used in this study are described in appendix A. Only the lowest elevation tilt of each TorNet sample was utilized for feature extraction.

1) TORNADO VORTEX SIGNATURE

The TVS is an operational algorithm within the WSR-88D ORPG. TVS is designed to detect tornadoes within radar data by analyzing various parameters such as radial velocity and reflectivity to identify rotation characteristics associated with tornadoes. In this study, TVS detections were obtained from the WSR-88D level-III archives. For each sample in TorNet, a binary classification of tornadic is made if a TVS detection lies within 30 km of the center of the chip. Otherwise, the sample is classified as nontornadic. This distance threshold of 30 km was an optimal hyperparameter identified during the training phase described in section 2.

2) LR

Logistic regression (LR) is a statistical and ML model used for binary classification tasks. It models the relationship between the input features and the probability of a sample being tornadic using the logistic function, which ensures that the output is between 0 and 1. The model then uses a threshold to make binary predictions. LR is a popular baseline because of its simplicity and efficiency. The class LogisticRegression from scikit-learn (Pedregosa et al. 2011) was used to fit this model. Optimal hyperparameters C (0.01) and class_weight (balanced) were identified during the training phase.

3) RF

Random forest (RF) is an ensemble learning algorithm used for both classification and regression tasks. It combines multiple decision trees to make more-accurate predictions. Each tree in the ensemble is trained on a random subset of the data and a random subset of the features. The binary predictions from individual trees are averaged to compute an average prediction score. RF is known for its high predictive accuracy, ability to handle large datasets with many features, and resistance to outliers and noise in the data. It is a popular choice in machine learning, especially for cases with tabular datasets. The class RandomForestClassifier from scikit-learn was used to fit this model. Optimal hyperparameters n_estimators (100), criterion (gini), max_depth (None), min_samples_split (2), min_samples_leaf (2), and class_weight (balanced) were identified during the training phase.

4) CNN

CNNs are deep learning models known for their exceptional ability to handle visual data for image classification tasks (He et al. 2016). The convolutional layers in a CNN apply filters to detect low-level features in the images, while deeper layers progressively learn more complex and abstract features. This hierarchical feature extraction allows CNNs to automatically learn relevant representations from raw pixel data, reducing the need for manual feature engineering. For this reason, the CNN baseline used in this work only takes raw radar imagery as input, and no additional processing or manual feature extraction is performed to augment the data. In particular, a derived azimuthal shear field is not provided to the CNN.

The CNN architecture developed for tornado detection in TorNet is provided in Fig. 3. The CNN takes as input 14 image channels that include two tilts for each of the six radar variables in TorNet, along with a binary mask of range-folded gates. In the interest of keeping this baseline simple and portable, only the last frame of each TorNet sample is provided to the model, although this architecture can be extended to take additional frames. Background gates in each channel are flagged, and the variables are each normalized to the range [0–1], with background and range-folded gates getting a value of −3.

After masking and normalization, the resulting 14-channel tensor is processed by a series of convolutional blocks, which apply nonlinear transformations and half the spatial resolution of the tensor through pooling. The number of output channels from block 1 is a hyperparameter, which is doubled by each subsequent block. These blocks are similar to convolutional blocks in the well-known Visual Geometry Group 19 (VGG19) architecture (Simonyan and Zisserman 2014), containing a sequence n_i ($i = 1, \ldots, N$) of 2D convolutional layers, followed by a 2×2 maximum pooling layer and a dropout layer (Srivastava et al. 2014). Instead of using traditional 2D convolutional layers, a variation called the CoordConv layer introduced in Liu et al. (2018) is utilized. The reason for this modification is that TorNet data samples are provided on a range-azimuth grid, and thus, there is no uniform spatial layout across rows and columns of the input grids (e.g., gates are spatially closer in the azimuthal dimension closer to the radar). This violates the spatial invariance of the convolution assumed in traditional image processing applications. To incorporate knowledge of the coordinate system into the convolution, the CoordConv concatenates additional layers to the input tensor prior to applying convolution, namely, a scaled radial coordinate r, along with its inverse r^{-1}. The CoordConv returns the output of the convolution, along with a (possibly downsampled) copy of the input coordinate tensor so that it can be used in downstream CoordConv layers. The additional inputs to the convolution allow the network to more easily recognize when data are closer to the radar, which is an important factor when recognizing tornadic signatures.

After being processed by the convolutional blocks, the data have a lowered spatial dimension, but a much-higher channel dimension. This channel dimension is reduced to 1 using two 1×1 convolutional layers, each with linear activation, which is processed with a sigmoid function to map values to [0, 1]. To create the final image classification, the maximum sigmoid activation within the sample is computed with a global maximum pooling layer, and it is passed to the loss function in addition to the true image labels (tornadic or nontornadic). The use of a global maximum pooling layer at the end of the network reduces the sensitivity of the location of tornadic features within the chip.

The CNN model described above was implemented using the Keras deep learning library (Chollet et al. 2015). The loss function used for classification is a binary cross-entropy loss function with label smoothing (Zhang et al. 2021). Label smoothing of 10%–20% was found to be effective at stabilizing training by making the loss less sensitive to weak or short-lived tornadoes that show little or no signature in radar data and to (possibly mislabeled) null cases that exhibit strong tornadic features. The CNN described above includes a number of tunable parameters, some of which were set manually due to the time it takes to train the CNN. These include the number of convolutional blocks ($N = 4$), the number of convolutions per block ($n_i = 2, 3, 3, 3$), the number of filters output by each block (starting with 48), and optimization settings like learning rate, loss weighting terms for tornadic samples, label smoothing, and number of epochs. Some parameters were tuned during the cross-validation procedure described in section 2 below. For the specific values of these, readers should consult the implementation provided in the paper's GitHub repository.

c. *Baseline results*

This section provides the training details and test performance of the baseline models on TorNet. We begin with an overview of the metrics, then describe the process for hyperparameter selection, and finally detail performance on the test set. Sample visualizations are also provided with classification labels given by the CNN model.

1) METRICS OF PERFORMANCE

With the exception of TVS, which is a binary indicator, each model outputs a continuous "likelihood" value $0 \leq p \leq 1$. For scoring, this value is converted to a binary indicator by choosing an appropriate threshold $0 \leq T \leq 1$. Given a threshold T, classifications over TorNet can be categorized and aggregated into a number of true positives (or "hits") $H(T)$, false positives (or "false alarms") $F(T)$, false negatives (or "misses") $M(T)$, and true negatives (or "correct rejections") $C(T)$. These counts are used to compute the following metrics which are all dependent on the choice of threshold T (which we omit writing for conciseness): accuracy: $\mathrm{ACC} = (H + C)/(H + C + M + F)$; true-positive rate (i.e., probability of detection, recall): $\mathrm{TPR} = H/(H + M)$; false-positive rate: $\mathrm{FPR} = F/(F + C)$; success rate (i.e., precision): $\mathrm{SR} = H/(H + F)$; and critical success index: $\mathrm{CSI} = H/(H + M + F)$.

To visualize several metrics across all potential thresholds, two kinds of performance curves are computed: i) receiver operating characteristic (ROC) curves, which display TPR on the y axis and FPR on the x axis, and ii) performance diagrams (PDs) (Roebber 2009), which also show TPR on the y axis, but instead show SR on the x axis. Points along these curves represent different choices of threshold T ranging from 0 (everything is a tornado) to 1 (nothing is a tornado). CSI can also be visualized in a PD since it is a nonlinear combination of both TPR and SR. For both types of curves, the area under the curve (AUC) can be computed to create a single metric of classifier performance which is not dependent on the choice of threshold. The area under the ROC curve will be abbreviated as AUC, while the area under (and to the left of) the performance diagram will be abbreviated as AUC-PD. To provide single values for ACC and CSI, the maximum value of these metrics is taken over all thresholds.

Relationships between output likelihood and frequency of occurrence can be assessed with the Brier score (BS) given by $\mathrm{BS} = (1/N)\sum_i (p_i - o_i)$, and Brier skill score (BSS) $= (\mathrm{BS}_{\mathrm{climo}} - \mathrm{BS})/\mathrm{BS}_{\mathrm{climo}}$, where p_i is the likelihood output of a classifier, o_i is the binary label, and $\mathrm{BS}_{\mathrm{climo}}$ is the BS of "climatology," which in the case of TorNet is the proportion of tornadic samples contained in the training data. The reliability of each classifier is visualized using attribute diagrams (Hsu and Murphy 1986).

TABLE 1. Optimal AUC scores of ML models considered from fivefold CV. The values of each fold represent AUC scores computed on the held-out validation set, and the final columns show the average and standard deviation across the fivefolds. The bold value signifies the highest average AUC score.

ML model	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean	Std
LR	0.8638	0.8489	0.8432	0.8536	0.8531	0.8525	0.0068
RF	0.8735	0.8624	0.8529	0.8615	0.8648	0.8630	0.0066
CNN	0.8893	0.8903	0.8693	0.8704	0.8884	**0.8815**	0.0096

2) CROSS VALIDATION

The ML models were trained in a two-step process:

1) Using fivefold cross validation (CV) to estimate optimal hyperparameters
2) Retraining the model on all training data using optimal hyperparameters

To perform a CV, the training partition of TorNet was divided into five nonoverlapping subsets, or folds, with 2013–14 in fold 1, 2015–16 in fold 2, 2017–18 in fold 3, 2019–20 in fold 4, and 2021–22 in fold 5. The testing partition was not utilized during the CV.

Each model was trained five times using fourfolds for training, with the remaining fold used for validation. Optimal parameter settings were selected by maximizing the AUC metric. For LR and RF models, the GridSearchCV class in scikit-learn was used to optimally tune hyperparameters for each split described above and AUC was used to select the best parameters for training the model on the entire dataset. Table 1 shows the performance of all models across the hold-out folds. The CNN model had the highest average AUC scores across the folds; however, it also had the largest standard deviation which suggests that the training procedure is most sensitive to initial random seeds, outliers in the training data, and possibly other factors.

To address the imbalance of labels in the datasets for training ML models, different class weights were assigned based on the label, category, and EF number of each sample. For LR and RF, the balanced weighting of the samples, where tornadic samples were weighted higher based on their proportion, showed a slight increase in the AUC score, but only by less than half a percent. For the CNN, the sample weights for confirmed, random, and warning categories were all treated as hyperparameters that were tuned on CV partitions. It was found that weighting tornadoes with an EF rating of 2 or higher with a weight of 2.0, warning samples with a weight of 0.5, and all others with a weight of 1.0 improved performance slightly, but again the highest gains were marginal.

For the CNN model, an important hyperparameter that was identified was the optimal number of epochs to train. As with all neural networks, training for too long may result in overfitting, which should be avoided. Figure 4 shows the cross-validation results across the five training folds for both the training partitions and held-out validation partitions. Two metrics are shown as a function of the training epoch, the training loss (cross entropy) on the top and the AUC on the bottom. In this case, the optimal number of training epochs was selected by maximizing the average AUC across the validation folds, as indicated in four. It can be seen that training for too many epochs reduces validation performance, suggesting that the CNN model overfits the training data for longer training times. Training of the CNN was done in a data-distributed manner utilizing two Volta 100 graphics processing units (GPUs) and took approximately 2 h per model that was trained.

3) PERFORMANCE COMPARISONS

Results are computed for three different views of the testing dataset:

1) All nulls (warnings and random) versus confirmed samples (NC): This method utilizes all the testing data. Because warnings are included, it is expected to have a higher number of false positives as a subset of these samples led to a human forecaster issuing a tornado warning.
2) Random versus confirmed samples (RC): This partition excludes the more difficult warnings cases and is expected to reflect performance in situations where a human forecaster did not think any tornadoes were present in the null samples.

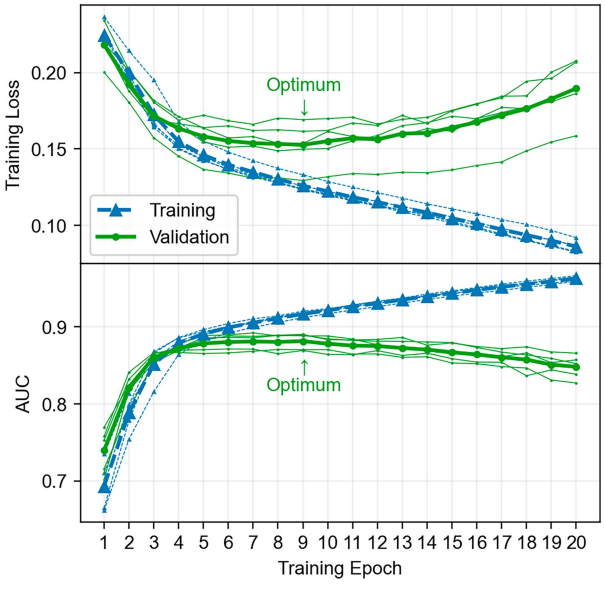


FIG. 4. Results of the fivefold CV for the CNN model. The plots show two metrics of classifier performance as a function of training epoch: (top) training loss and (bottom) AUC. The thin and dashed lines represent the performance of different train/validation partitions, and the solid lines are the average over all folds. In this case, the optimal number of training epochs (9) was identified by finding the maximum AUC on the validation set.

TABLE 2. Performance results across baseline models on different views of the TorNet test set. The values represent the average over each model trained with five different random seeds, with standard deviations in parentheses. The values in bold signify the best-performing model for each metric.

Baseline	ACC	AUC	AUC-PD	CSI
		(NC) All nulls versus confirmed		
NoTornado	0.9367 (0)	0.5000 (0)	0.0633 (0)	0.0000 (0)
TVS	0.9260 (0)	0.6308 (0)	0.1583 (0)	0.2002 (0)
LR	0.9411 (0.0×10^{0})	0.8498 (0.0×10^{0})	0.3806 (0.0×10^{0})	0.2667 (0.0×10^{0})
RF	0.9477 (1.3×10^{-4})	0.8557 (1.1×10^{-3})	0.4732 (3.3×10^{-3})	0.3066 (8.7×10^{-4})
CNN	**0.9505** (7.8×10^{-4})	**0.8760** (5.3×10^{-3})	**0.5294** (8.8×10^{-3})	**0.3487** (6.9×10^{-3})
		(RC) Random versus confirmed		
NoTornado	0.9041 (0)	0.5000 (0)	0.0959 (0)	0.0000 (0)
TVS	0.9272 (0)	0.6436 (0)	0.3163 (0)	0.2783 (0)
LR	0.9356 (0.0×10^{0})	0.9161 (0.0×10^{0})	0.6562 (0.0×10^{0})	0.4538 (0.0×10^{0})
RF	0.9404 (1.7×10^{-4})	0.9090 (6.5×10^{-4})	0.6969 (1.2×10^{-3})	0.4598 (1.7×10^{-3})
CNN	**0.9471** (1.5×10^{-3})	**0.9245** (5.6×10^{-3})	**0.7436** (1.1×10^{-2})	**0.5163** (1.3×10^{-2})
		(WC) Warnings versus confirmed		
NoTornado	0.8432 (0)	0.5000 (0)	0.1568 (0)	0.0000 (0)
TVS	0.8247 (0)	0.6082 (0)	0.2329 (0)	0.2076 (0)
LR	0.8569 (0.0×10^{0})	0.7335 (0.0×10^{0})	0.4341 (0.0×10^{0})	0.2880 (0.0×10^{0})
RF	0.8714 (3.8×10^{-4})	0.7611 (1.2×10^{-3})	0.5148 (2.0×10^{-3})	0.3242 (2.8×10^{-3})
CNN	**0.8797** (2.1×10^{-3})	**0.7910** (4.9×10^{-3})	**0.5698** (7.3×10^{-3})	**0.3617** (7.1×10^{-3})

3) Warnings versus confirmed samples (WC): This represents the most difficult view of the data from a classification standpoint, as nontornadic samples are taken from within tornado warning polygons and thus are likely to exhibit tornadic features.

The performance metrics computed for each model and view are shown in Table 2. For each model, this table provides the average performance of the model trained using the entire training set, averaged over five random seeds. These random seeds determine weight initialization, dropout, sampling, and other steps performed in training algorithms that are based on random numbers. The standard deviation of each metric across the seeds is also shown in parentheses to give a measure of the sensitivity of each model to randomness in training. For ACC, a model that always outputs "no tornado" obtains an accuracy of 93% in view (NC), which reflects the imbalance of the dataset (and emphasizes that ACC is not a great metric for imbalanced datasets). The ML models trained on TorNet all show noticeable gains over (the non-ML) TVS across all metrics considered. Of the ML models, CNN is the top performer on average as was seen during the CV. It is notable that the CNN also carries a higher standard deviation across random seeds, sometimes by an order of magnitude relative to the RF. This can be explained by the increased use of randomness in the CNN training process. Certain strategies, like averaging over ensembles of trained models, or employing more regularization during training, could potentially help alleviate this.

For view (RC), scores improved overall compared to view (NC) due to the removal of the warning category, which confirms that warning samples are indeed more difficult from a detection standpoint. The ordering of model performance remains virtually the same, with some exceptions (e.g., LR having a higher AUC than RF). For view (WC), ACC and AUC scores decreased compared to the other cases due to the removal of the random category, while the AUC-PD and CSI scores actually increased slightly relative to (NC). This can be explained by the fact that the majority of samples in the random category result in correct rejections, which do not factor into the calculation of AUC-PD or CSI, but have a large influence on ACC and AUC.

ROC curves and PD for partition (NC) are shown in Fig. 5. The ROC curve is shown on the left, and PD on the right, with each displaying the performance of the four baseline models, along with the no tornado model (black dashed line). In both plots, the TVS baseline is indicated by a single point since it is a deterministic model that does not depend on any threshold. The point along each curve corresponds to the likelihood threshold that yields the maximum CSI shown in Table 2.

In the ROC curve, the TVS baseline falls toward the bottom left portion of the plot, indicating that this algorithm targeted a low FPR (at the expense of also showing a low TPR). In comparison with the ML baselines, the TVS underperforms in regards to both TPR and FPR for a range of different thresholds. Among the ML models, the CNN offers the best overall AUC, and it shows notable separation from other baselines in the midthresholds, but performs similarly to the RF model for higher thresholds. A similar trend is noticed in the PD, with the TVS underperforming all ML baselines. Again, the CNN model shows the highest performance, followed by the RF and LR models. ROC curves and PD for partitions (RC) and (WC) of the test set are included in appendix B.

4) ATTRIBUTES DIAGRAMS

Attribute diagrams are provided for the three ML baselines in Fig. 6. These diagrams show the reliability of classifier likelihood (x axis) compared to the probability of tornado occurrence conditioned on classifier output (y axis). Each curve uses 10 bins along the x axis to approximate these conditional probabilities. It is important to note that the term "probability" in

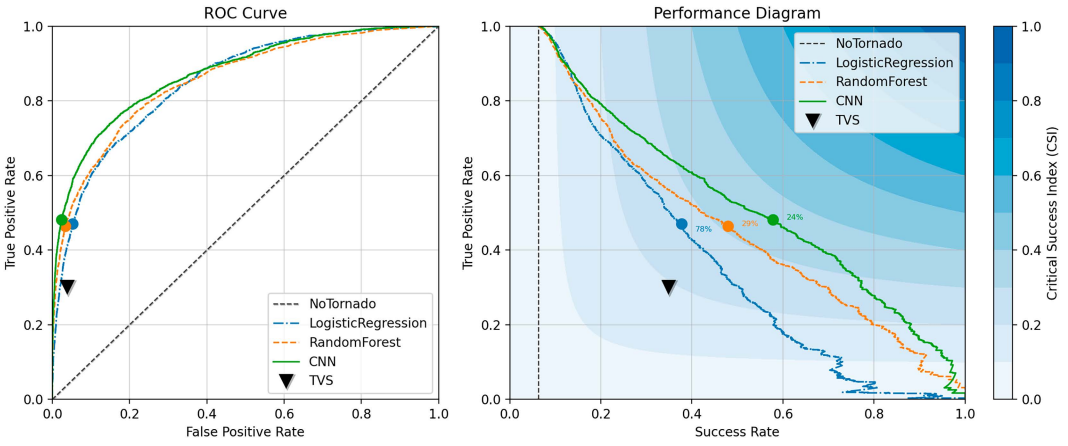


FIG. 5. (left) ROC curve and (right) PD for the entire test set (confirmed versus all nulls). In these plots, four baseline models are compared: TVS (which is represented as a single point in this space since it is a deterministic output), LR, RF, and CNN detectors. All ML models considerably outperform the TVS algorithm. In both the ROC and PDs, the CNN model showed the greatest AUC. The point along each curve corresponds to the likelihood threshold that maximizes CSI. The value of each threshold is indicated next to each curve in the PD.

this context refers to the distribution of samples from the TorNet dataset, which is biased toward tornado observations. Nevertheless, it is still useful to examine how tornado likelihoods are related to observed frequencies within the dataset. The diagonal 1–1 line in each attribute diagram denotes a perfectly reliable classifier. The vertical and horizontal dashed lines represent the climatology line and the no-resolution line. The shaded region denotes the regions where the skill of the classifier has a higher BS than climatology (Haynes et al. 2023). The histograms inserted into each attribute diagram show the distribution of likelihoods generated on the test set.

All reliability curves are monotonically increasing across the entire range of 0–1, suggesting all classifiers trained on TorNet provide some resolution in their predictions. The RF and CNN models produce more reliable likelihoods compared to LR, as these curves lie close to the 1–1 line, while the LR model is overconfident. The CNN model shows the best reliability and also achieves the highest BSS of 0.314.

5) VISUALIZATIONS

This section provides visualizations of various samples in TorNet accompanied by the likelihood p output by the CNN classifier. All samples visualized in this section are selected from the test set and were unseen during model training. Four visualization candidates were chosen across low (0%–15%), mid (15%–50%), and high (50%–100%) probability ranges to help develop intuition about how the classifier responds to certain visual features in the data. Using the maximum CSI threshold $T_* = 0.24$ identified in Fig. 5, we select data from four categories: i) hits, where the sample is a confirmed tornado and $p > T_*$; ii) correct rejections, where there is no confirmed tornado and $p < T_*$; iii) misses, where this is a confirmed tornado and $p \leq T_*$; and iv) false alarms, where there is no confirmed tornado and $p \geq T_*$. Only the

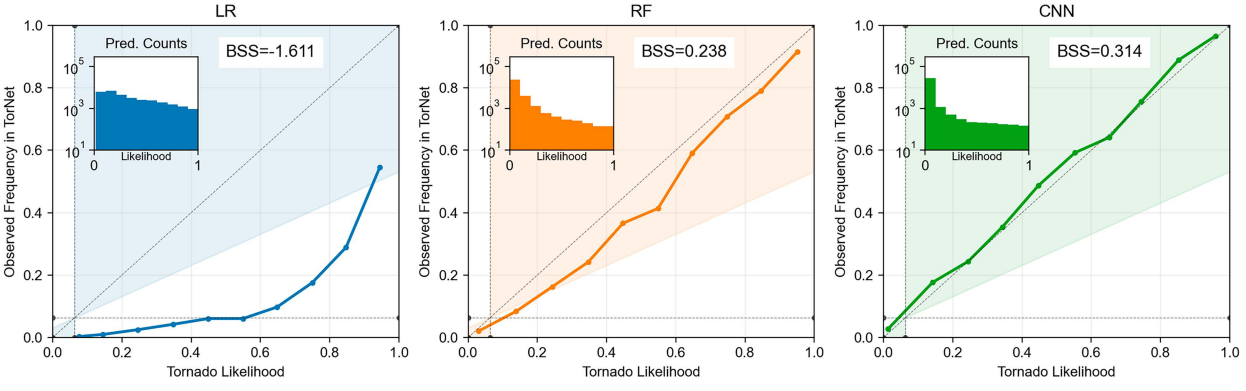


FIG. 6. Attribute diagram for the three ML classifiers. These curves show the reliability of each classifier's likelihood (x axis) compared to the probability of tornado occurrence conditioned on classifier output (y axis). Of the three ML baselines, (right) the CNN model shows the best reliability and highest BSS.

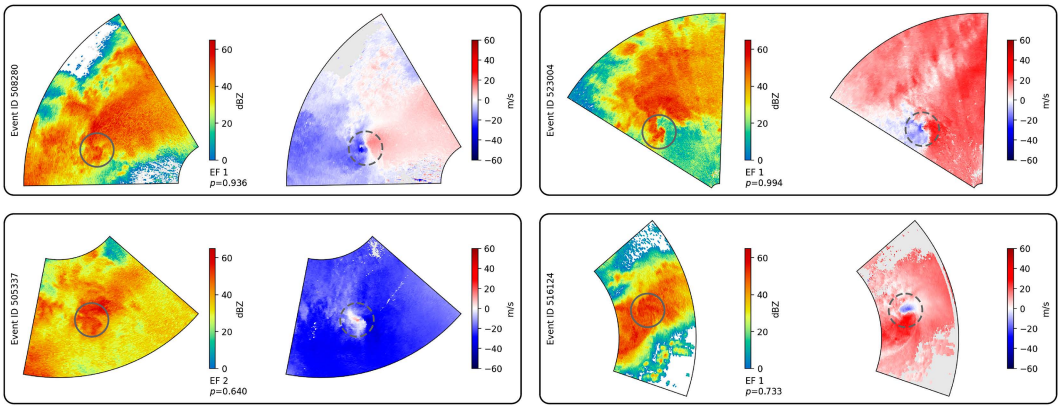


FIG. 7. Hits: example tornadic cases that the CNN model correctly classified. Hook echoes in the reflectivity factor field are annotated as solid gray circles, and velocity couplets in the radial velocity field are annotated as dashed gray circles.

reflectivity factor and radial velocity are shown; however, the CNN model utilized all variables in the TorNet samples.

The hits shown in Fig. 7 show four samples that were correctly classified. Each of these shows common radar characteristics of tornado signatures, including hook echoes in the reflectivity factor field (annotated as solid gray circles) and velocity couplets in the radial velocity field (annotated as dashed gray circles). Subjectively, the probabilities assigned with these samples seem to align well with the strength of the radar signals. The correct rejections in Fig. 8 also demonstrate examples of obvious nontornadoes with low assigned probability values. It is interesting to note in the cases along the bottom, where the model assigns p close to the threshold of 0.24, likely due to noticeable azimuthal shear observed in radial velocity.

The misses and false alarms in Figs. 9 and 10 depict incorrect classifications. The misses shown are EF0 and EF1 tornadoes that do not exhibit extremely strong or obvious hook echoes or rotation signatures. In the cases along the bottom of Fig. 9, the CNN did assign probabilities greater than 10%, likely because of weak rotation signatures and/or the presence of range folding (which would mask signal and increase uncertainty). The selected false alarm cases show four warning cases from TorNet that are assigned high likelihoods, despite no tornado being confirmed.

4. Detection in full radar scans

This section demonstrates how ML models trained using TorNet to classify tornado signatures in individual samples can be adapted for real-time monitoring of tornadoes on full-scale WSR-88D data. While any of the ML baselines described in section 3 could be adapted for this, the CNN model is highlighted in this section not only because it shows the highest overall skill but also because CNNs are extremely efficient at processing large imagery, with inference on a full WSR-88D scan taking approximately 1 s on CPU and fractions of a second on GPU. To adapt the CNN for inference on full-sized imagery, the global maximum pooling layer that computes the maximum likelihood over the entire chip is removed. With this final layer removed, the CNN is fully convolutional (meaning it can be applied to input images of arbitrary size) and outputs a single-channel likelihood field that is sized proportionally to the input grid. Since $N = 4$

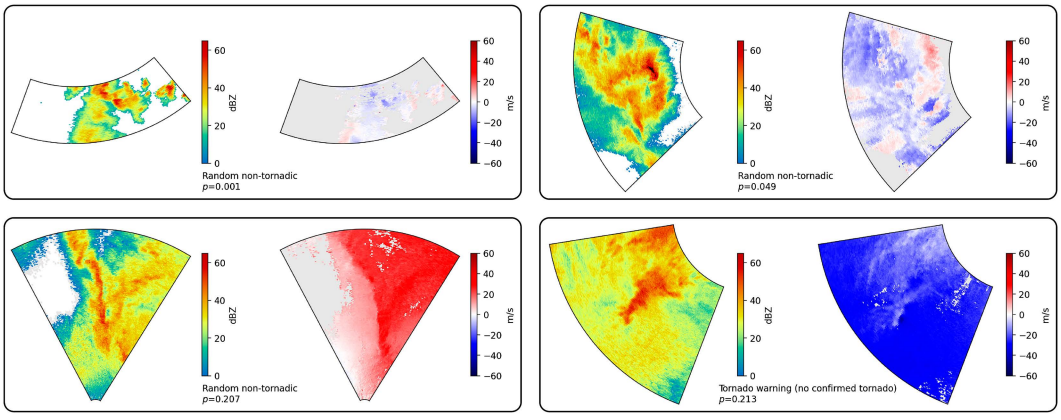


FIG. 8. Correct rejections: example nontornadic cases that the CNN model correctly classified.

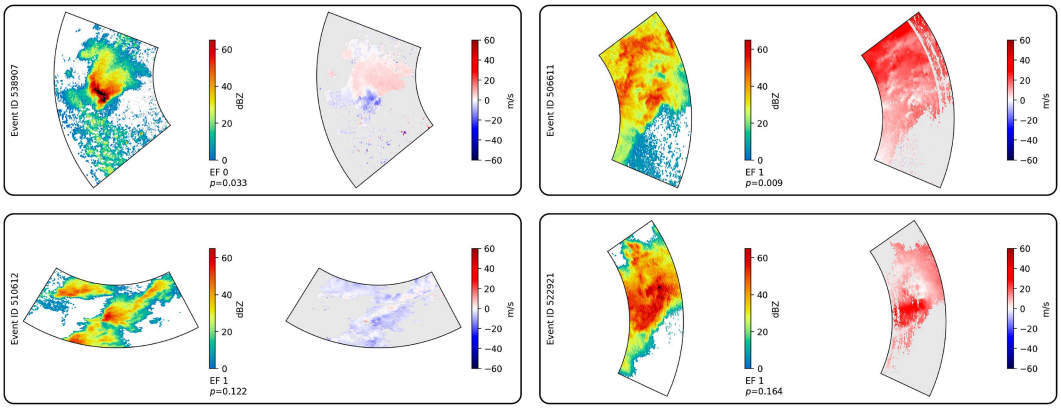


FIG. 9. Misses: example tornadic cases that the CNN model incorrectly classified as nontornadic.

processing blocks containing 2×2 maximum pooling layers were utilized in the baseline, the spatial dimension of the output likelihood will be 1/16 the size of the inputs. This output can be downscaled to the input image size using bilinear interpolation and resampled to Cartesian coordinates to produce a map of tornado likelihoods. To ensure proper spatial alignment after this downscaling, inputs should be padded so that their shapes are evenly divisible by 2^N. This section applies this process to selected case studies to demonstrate how the trained CNN model performs on entire storm episodes in the TorNet test set.

The first case study considered is a tornado outbreak that occurred in the lower Mississippi (MS) River and Tennessee (TN) River Valleys on 28 April 2014 starting at approximately 1800 UTC, with storm episode ID 84120. The storm system, driven by a strong jet stream and deep surface cyclone, caused damaging wind, large hail, and multiple tornadoes across the NWS Jackson, Mississippi forecast area. The most destructive tornado was an EF4, covering 34.3 mi and causing 10 fatalities. In total, 21 tornadoes were confirmed in the region, ranging from EF0 to EF4, with several counties experiencing significant damage. The confirmed tornadoes from this outbreak resulted in 70 TorNet radar image samples.

A single time stamp from this episode is shown in Fig. 11. The left two panels depict DBZ and VEL, respectively, at 2232:30 UTC on 28 April 2014, resampled to Cartesian coordinates using a bilinear interpolation. At this time, a strong tornado signature emphasized by a hook echo in the reflectivity factor and a strong couplet in radial velocity is indicated by the black circle. The rightmost panel provides an output of the CNN model, which creates a likelihood at each grid cell. In this case, the CNN produces a high likelihood that lines up with the features observed in the reflectivity factor and radial velocity. Moreover, the signal is localized in the region around the tornado, with no other high likelihoods present in other areas of the scene. The shape of the high-likelihood region appears slightly stretched relative to the azimuth dimension, likely due to the kernels used in the CNN architecture being applied to the data in polar coordinates.

To visualize performance over an entire storm episode, the likelihood shown in the right panel of Fig. 11 can be aggregated by computing the maximum observed likelihood over all WSR-88D observations in either time or space. For example, when spatial likelihood maps are aggregated over time, they depict tracks of high likelihood that can be compared to confirmed tornado track(s) (black lines), similar to how

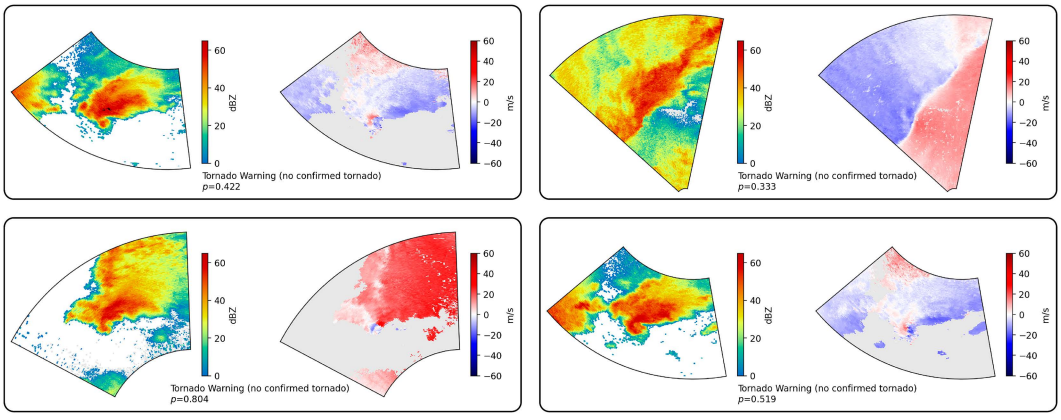


FIG. 10. False alarms: example warning cases that the CNN model classified as tornadic.

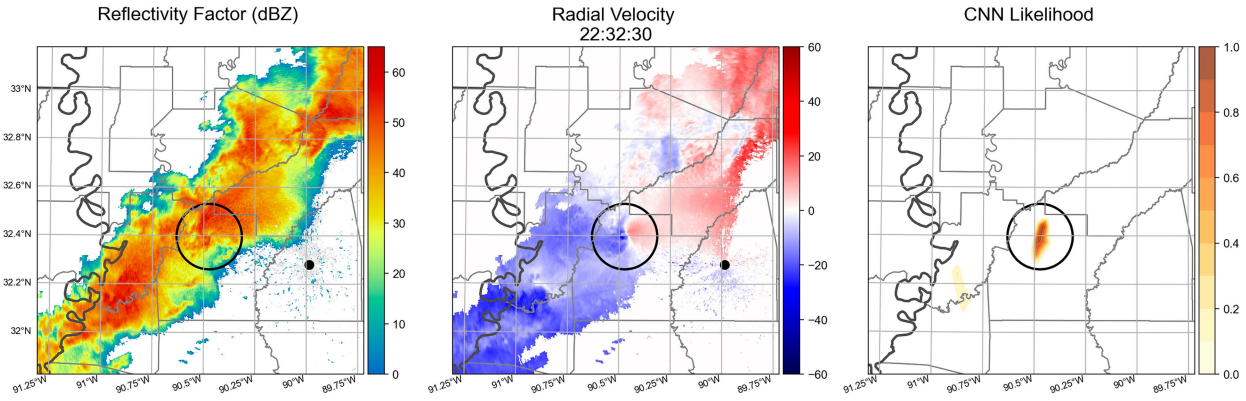


FIG. 11. Detection of a tornado near KDGX, Dodge City, Kansas, on 28 Apr 2014 using the CNN model trained on TorNet. (left) Reflectivity factor and (center) radial velocity. The circle in the middle of the images indicates the location of a confirmed tornado. The CNN model described in section 3 was applied to the entire radar scene, and the associated likelihood field is shown in the right panel, with a strongly elevated signal corresponding to the location of the tornado and low likelihood in areas away from the tornado.

azimuthal shear is aggregated in Mahalik et al. (2019). This is a useful spatial verification technique to see identified tornadoes, misses, and/or false alarms. Alternatively, the likelihoods for each time can be aggregated over a region of interest to create a time series.

For example, aggregated likelihoods computed between 2127 UTC 28 April and 0019 UTC 29 April are shown in the left panel of Fig. 12. All confirmed tornadoes within this time range are denoted with black lines. Most of the tornado tracks in this time range correspond with a region of high likelihood, with weaker signals being seen in the rightmost track corresponding to a tornadic cell that passed directly over the radar. This situation likely created difficulty for the CNN algorithm, which is limited by the kernel sizes of the CNN kernels and thus was not able to "see" the entire tornado represented in polar coordinates (a limitation of this approach when very close to the radar). Moreover, regions not near a tornado track exhibit lower likelihoods. The red polygons correspond

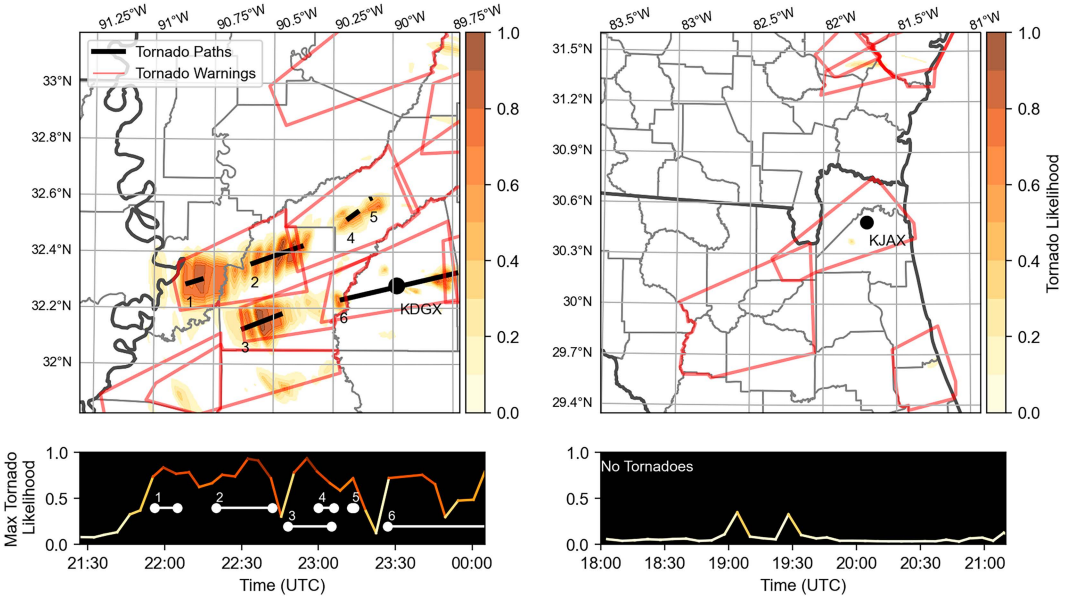


FIG. 12. Examples of tornado likelihoods aggregated over two selected storm episodes. (left) A tornado outbreak starting 28 Apr 2014 near KDGX. The black lines represent confirmed tornado tracks. CNN tornado likelihoods were aggregated over a 3-h period and are shown as shaded areas. The maximum likelihood aggregated over the region for each time is plotted along the bottom. In this case, the tracks of high likelihood correspond to the confirmed tornado tracks. (right) A similar visualization starting 3 Mar 2018 near KJAX with no confirmed tornadoes. In this case, the likelihood remains low for the entirety of the time interval considered. Tornado warnings were issued for this case, and they are shown as red polygons.

to tornado warnings issued during this time period. Note that regions of low–moderate likelihoods in the southern part of the image are contained in these polygons.

The right panel of Fig. 12 shows another storm episode near KJAX, Jacksonville, Florida, starting on 3 March 2018. Unlike the previous case, this episode did not exhibit any confirmed tornadoes, although tornado warnings were issued. In this case, the aggregated likelihood map shows overall low probabilities everywhere in the region, with the only occurrences happening in areas where tornado warnings were issued.

5. Discussion of limitations

TorNet was created to provide an easily accessible benchmark dataset for the study of tornadoes. However, like other well-known benchmark datasets in the machine learning literature (e.g., Northcutt et al. 2021; Vasudevan et al. 2022), TorNet has known limitations related to potential labeling errors, biases, and other factors that users should be aware of when developing and validating models. Below, we discuss some of these limitations, along with possible mitigation strategies.

a. Unconfirmed tornadoes

TorNet labels are based on the existence of confirmed tornado reports in the NSED. These confirmations are made by trained spotters who combine information from radar data and ground surveys of wind damage (NWS 2024). While it is reasonable to assume that confirmed tornado reports correspond to actual tornadoes, there is a possibility that unconfirmed tornadoes were labeled as a null event in TorNet (most likely in the warning category). For training ML models, there are a number of strategies one can take to account for the possibility of incorrect labels (Natarajan et al. 2013; Song et al. 2023). When training the CNN baseline, a lower weighting was assigned to the warning class, and label smoothing was applied to reduce the impact of incorrect labels on the loss function. Many other strategies could be applied and may be explored in future work.

For model testing, TorNet provides a convenient means for computing standard test metrics across different classifiers. However, mislabeled samples can impact the generalizability of these metrics to other datasets. In this work, metrics were computed across multiple views of the test data as a way of showing robustness to different levels of potential label noise. More thorough uncertainty estimates of testing metrics can be computed by assuming nonzero prior over label noise, and inferring Bayesian posteriors over test metrics, as described in Su (2024).

b. Limited temporal resolution

Many tornadoes are very short lived. Within TorNet, 44% of tornadic samples are associated with tornadoes with a duration of 5 min or less, and 13% of samples have a minimum recorded duration of 1 min in the NSED. These represent a nontrivial subset of the data, and while one can be confident that these tornadoes did occur, it is highly possible that the scan rate of the NEXRAD radar did not adequately capture these short-lived tornadoes. TorNet includes scans taken within at most 5 min of the recorded tornado event. Therefore, it is possible that radar data are sampled immediately before the tornado forms or after it dissipates. These cases were intentionally not filtered to allow models to capture the limitations of radar sampling and may also be useful for understanding precursors of tornadogenesis. The start and end times of each tornadic event in the NSED are included in TorNet to help researchers account for the impact of short-lived tornadoes on performance.

c. Biased sampling

The sampling method in TorNet oversamples images from tornadic events and tornado warnings, leading to a skewed representation of the underlying probabilistic distribution of these events. Users of the dataset should be aware of this bias, which was intentionally introduced to expose machine learning models to rare events. Future work will explore the use of larger case studies, like those presented in section 4, as a way to calibrate metrics so they are more transferable to how an algorithm trained using TorNet would perform in an operational setting.

6. Conclusions

This study introduces a public benchmark dataset called TorNet for tornado detection and prediction. Benchmark datasets are a critical component of AI/ML fields, and meteorology is no different. A combination of data sources, including the NCEI Storm Events Database and level-II/III WSR-88D data at full resolution and multiple tilts, is used to create hundreds of thousands of samples focused on relevant tornadic and nontornadic scenes. This dataset was curated and labeled for ML purposes, although such a dataset may be useful in many other ways, including large-scale case studies, automation, and algorithm development and testing. The dataset is made freely available in a format that can be easily parsed and filtered and is straightforward to work with in a user programming environment of choice.

A number of ML baselines were developed and compared for the important problem of tornado detection. Baseline results showed that the performance of ML models trained on TorNet exceeds that of an operational baseline algorithm (TVS). The best-performing (and most complex) model was a CNN, a DL model that utilized full-resolution imagery directly and did not need features to be defined and extracted beforehand. This model was also demonstrated in selected case studies to show how ML models trained on TorNet can be adopted for real-time detection and tracking of tornadoes in full-sized radar imagery.

We hope that in future studies, the dataset leads to innovation in this research area that will exceed and/or build upon the baselines presented in this work. Future directions include the possible addition of more data sources, such as more tilts/time frames of radar data, various satellite imaging channels, lightning from the global lightning mapper (GLM) aboard the Geostationary Operational Environmental Satellite (GOES) constellation, and NWP model output. Multiple

studies show the benefits of ML for fusing multiple modalities of data, leading to our hope that continuing to build TorNet will yield increasingly interesting and impressive results. Taking the need to develop this dataset from scratch out of the time budget for researchers is anticipated to allow for new and improved AI/ML techniques for this problem, including those who may not be domain experts. Additionally, through the public release and open-source nature of the dataset, it is hoped that continued growth and improvement driven by the community will only strengthen its impact. We hope to see this approach taken in many contexts within the field of meteorology in the coming years.

Going forward with this specific dataset, the problem of tornado prediction is an obvious next step. As mentioned previously, the TorNet dataset contains data before tornadogenesis, allowing for the potential to predict tornadoes at given temporal thresholds. While the utility of this dataset to the prediction problem has not yet been fully explored, it is the immediate next step for the ongoing research. When combined with additional data modalities, it is hoped that the viability for a tornado prediction algorithm will be sufficient.

Acknowledgments. The authors would also like to thank the anonymous reviewers for their valuable insights and suggestions. DISTRIBUTION STATEMENT A. Approved for public release: distribution unlimited. This material is based on work supported by the Department of the Air Force under Air Force Contract No. FA8702-15-D-0001. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Department of the Air Force. The research was also sponsored by the United States Air Force Research Laboratory and the Department of the Air Force Artificial Intelligence Accelerator and was accomplished under Cooperative Agreement Number FA8750-19-2-1000. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Department of the Air Force or the U.S. government. The U.S. government is authorized to reproduce and distribute reprints for government purposes notwithstanding any copyright notation herein.

Data availability statement. Instructions for downloading TorNet can be found at https://github.com/mit-ll/tornet, along with Python code for recreating the CNN baseline. Archives of level-II WSR-88D data used for building TorNet can be found in the AWS Open Data Registry at https://registry.opendata.aws/noaa-nexrad/. Level-III archives for the WSR-88D network were obtained from Google Cloud Storage at https://console.cloud.google.com/storage/browser/gcp-public-data-nexrad-l3. The NSED can be downloaded from NCDC at https://www.ncdc.noaa.gov/stormevents/. Tornado warning polygons can be obtained from Iowa State University's archive found at https://mesonet.agron.iastate.edu/vtec/search.php.

APPENDIX A

Feature Extraction

This section describes the feature extraction process for non-DL methods covered in section 3b. An overview of feature extraction is shown in Fig. A1. First, the reflectivity factor and radial velocity channels are combined to create an azimuthal shear field (AzShear). AzShear is created by first isolating regions with reflectivity above a threshold of 20 dB*Z*. Within these areas, the azimuthal gradient of radial velocity is computed using the linear least squares

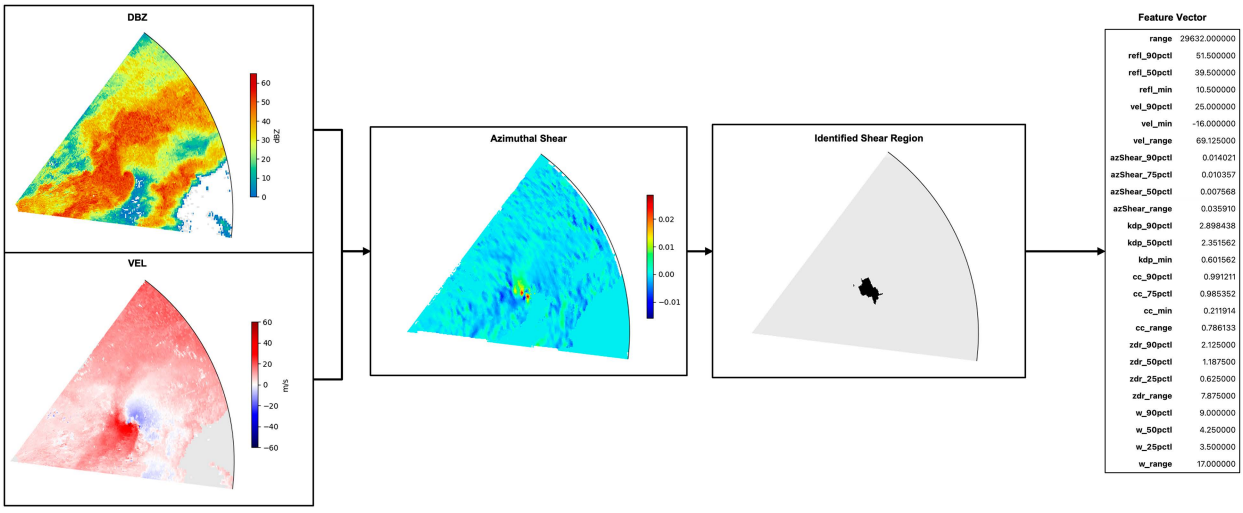


FIG. A1. Process for extracting features from data chips for use in non-DL models. Tilts of DBZ and VEL (left images) are combined to compute azimuthal shear (second image from the left). Regions of high shear are identified using an object identification algorithm (third image from left). Within these identified objects, a feature vector is computed across a number of different radar values.

derivative (LLSD) method as described in Smith and Elmore (2004) using a square kernel size of 1.5 km. Next, a threshold of 0.006 s^{-1} is applied to the AzShear field, and a region extraction algorithm is applied to isolate regions of high AzShear larger than a minimal size threshold. An image close operation (Dougherty 1992) is performed on the identified regions, and if any regions remain, the one within the chip containing the highest value of AzShear is selected. If no regions remain, then a circular region with a radius of 5 km in the center of the chip is used instead.

To extract a feature vector, a number of statistics listed in the table in the right portion of Fig. A1 are measured from within the selected region of high AzShear across the six radar variables from each TorNet sample and the derived AzShear field. Statistics computed for each variable include the 90th, 75th, 50th, and 25th percentiles, minimum value, and range of values. In addition, the chip's range from the radar is also included as a feature. Only the lowest tilt was used for feature extraction.

APPENDIX B

Additional Performance Results

In addition to the curves shown in Fig. 5, ROC curves and PD were also computed for two additional views of the test sets (RC) and (WC) described in section 3.

Results for the test partition (RC), where only the random category is used for null cases, are shown in Fig. B1. Compared to the results in Fig. 5, it is clear that performance increases across all models. This is due to the fact that warning samples contain cases that are harder for the models to classify correctly. With this partition, the logistic regression and random forest models show similar performance and still underperform the CNN for most thresholds. The TVS again shows a low FPR/low TPR and shows worse performance than the ML counterparts.

Results for the final partition (WC) are shown in Fig. B2. Compared to the ROC curves in partitions (NC) and (RC), the results for this partition are the worst overall. This is to be expected, since the warning categories contain imagery

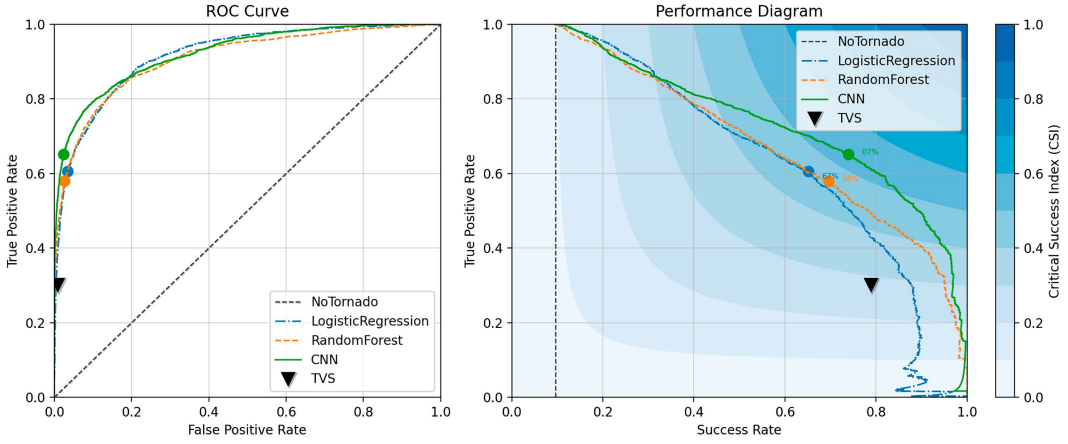


FIG. B1. As in Fig. 5, but only the random category is used for nontornadoes.

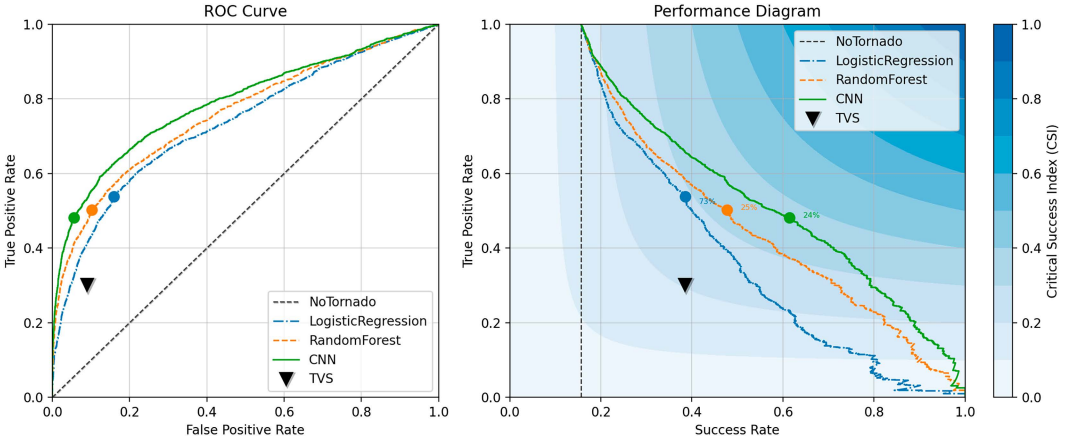


FIG. B2. As in Fig. 5, but only the warnings category is used for nontornadoes.

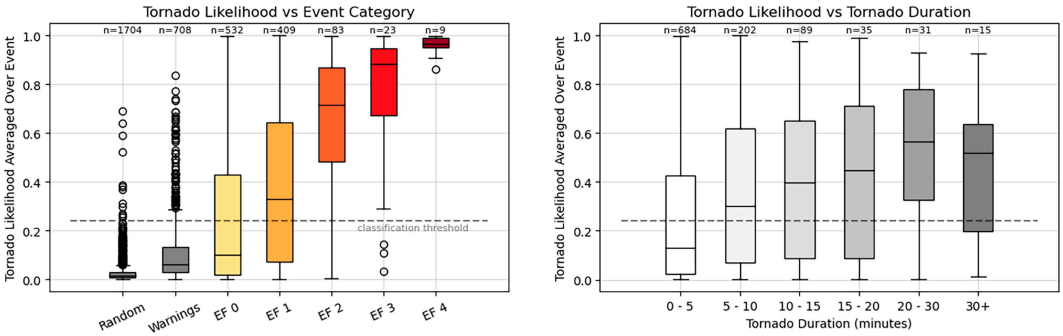


FIG. B3. Distributions of CNN likelihoods conditioned on (left) EF rating and (right) tornado duration. The distributions reflect the average likelihood computed over storm events in TorNet.

that led human forecasters to issue tornado warnings. Not surprisingly, this results in the classification models all having higher FPR compared to the other partitions. Also of note is that the PD bears similarity to partition (NC), implying that most of the false positives (which drives SR) observed in partition (NC) came from the warnings category. Within this partition, we again see the CNN outperforming other baselines and TVS underperforming all ML baselines considered.

The performance of the CNN classifier was also analyzed across different EF ratings and is summarized in the left panel of Fig. B3. The bars reflect the distribution of tornado likelihoods averaged over storm events in the TorNet test set. A dashed horizontal line indicates the threshold of 0.24 identified in Fig. 5. High likelihoods are generated, on average, for tornadoes with higher EF ratings. Conversely, many EF0 samples fall below the detection threshold, due likely to the short duration and weak signature in radar data of these tornadoes. Samples in the warning category exhibit higher likelihoods compared to samples in the random category.

A similar analysis was performed based on tornado duration in the right panel of Fig. B3. This shows that for longer-lived tornadoes, the average likelihood produced by the CNN is higher. Most tornadoes are short lived, as indicated by the leftmost box in this diagram, and these tornadoes generate lower likelihood values on average.

APPENDIX C

Radar Variables

In this study, several radar variables are utilized to build a large, curated dataset for tornado detection. Figure C1 provides an overview of the key radar variables employed, which include both standard and polarimetric estimates. Standard radar variables, such as horizontal reflectivity factor (DBZ), radial velocity (VEL), and spectrum width (WIDTH), are used for assessing the concentration, motion, and variability of atmospheric scatterers. DBZ provides insights into the density and size distribution of particles within a given volume, while VEL offers information on the velocity and directivity of motion. WIDTH estimates the distribution of radial velocities, indicating the extent of differential motion within the radar volume.

In addition to these standard variables, polarimetric radar variables offer enhanced capabilities for distinguishing between different types of scatterers and improving precipitation estimates. Differential reflectivity (ZDR) estimates the ratio of horizontally to vertically polarized reflectivity factors, providing valuable information on particle shape and orientation. The copolar correlation coefficient (RHOHV) indicates the correlation between horizontally and vertically polarized returns, aiding in the identification of meteorological and nonmeteorological scatterers. Last, specific differential phase (KDP) is used to estimate liquid water content and rainfall rate by examining the phase shift between horizontally and vertically polarized waves.

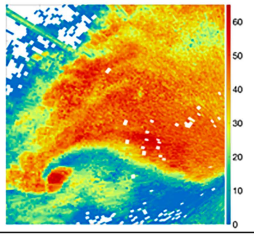
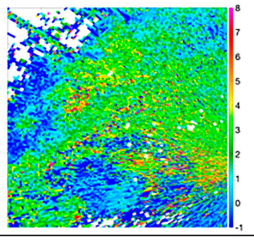
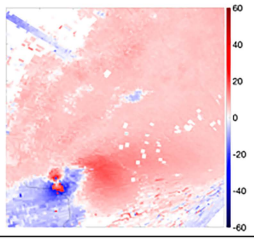
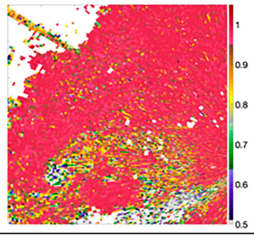
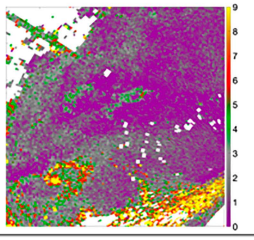
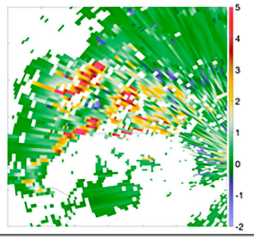
Standard Variables		Polarimetric Variables	
Radar Variable	**Example**	**Radar Variable**	**Example**
Horizontal Reflectivity Factor – DBZ (dBZ): The equivalent logarithmic radar backscattered power within the resolution volume; primarily indicative of the size and concentration distribution of scatterers.		**Differential Reflectivity – ZDR (dB)**: The logarithmic ratio of the horizontally to vertically polarized reflectivity factors; provides insights into scatterer shape, size, and orientation.	
Radial Velocity – VEL (m s^{-1}): The mean Doppler velocity of a volume of scatterers along the antenna beam axis; for assessing wind speed and direction.		**Co-Polar Correlation Coefficient – RHOHV (unitless)**: The correlation between the horizontally and vertically polarized channels; aides in the identification of hydrometeor types and the presence of non-meteorological scatterers.	
Spectrum Width – WIDTH (m s^{-1}): A measure of the dispersion of radial velocities within a volume; reflects the broadening of the Doppler spectrum due to differential motion.		**Specific Differential Phase – KDP (deg km^{-1})**: The range derivative of the differential phase shift between the horizontally and vertically polarized channels; a tool for estimating liquid water content and rainfall rate.	

FIG. C1. Descriptions and examples of the radar variables used in this study. Examples are from 20 May 2013 at 2003:56 UTC at the KTLX WSR-88D in Oklahoma City, Oklahoma.

REFERENCES

Ali, H., M. N. M. Salleh, R. Saedudin, K. Hussain, and M. F. Mushtaq, 2019: Imbalance class problems in data mining: A review. *Indones. J. Electr. Eng. Comput. Sci.*, **14**, 1560–1571, https://doi.org/10.11591/ijeecs.v14.i3.pp1552-1563.

Betancourt, C., T. Stomberg, R. Roscher, M. G. Schultz, and S. Stadtler, 2021: AQ-Bench: A benchmark dataset for machine learning on global air quality metrics. *Earth Syst. Sci. Data*, **13**, 3013–3033, https://doi.org/10.5194/essd-13-3013-2021.

Bishop, C. M., 2007: *Pattern Recognition and Machine Learning*. Information Science and Statistics, Vol. 44, Springer, 758 pp., https://doi.org/10.5860/CHOICE.44-5091.

Brown, R. A., and V. T. Wood, 2012: The tornadic vortex signature: An update. *Wea. Forecasting*, **27**, 525–530, https://doi.org/10.1175/WAF-D-11-00111.1.

Chase, R. J., D. R. Harrison, A. Burke, G. M. Lackmann, and A. McGovern, 2022: A machine learning tutorial for operational meteorology, Part I: Traditional machine learning. arXiv, 2204.07492v2, https://doi.org/10.48550/arxiv.2204.07492.

——, ——, G. M. Lackmann, and A. McGovern, 2023: A machine learning tutorial for operational meteorology. Part II: Neural networks and deep learning. *Wea. Forecasting*, **38**, 1271–1293, https://doi.org/10.1175/WAF-D-22-0187.1.

Chollet, F., and Coauthors, 2015: Keras. https://keras.io.

Cintineo, J. L., and Coauthors, 2018: The NOAA/CIMSS ProbSevere Model: Incorporation of total lightning and validation. *Wea. Forecasting*, **33**, 331–345, https://doi.org/10.1175/WAF-D-17-0099.1.

——, M. J. Pavolonis, J. M. Sieglaff, L. Cronce, and J. Brunner, 2020: NOAA ProbSevere v2.0—ProbHail, ProbWind, and ProbTor. *Wea. Forecasting*, **35**, 1523–1543, https://doi.org/10.1175/WAF-D-19-0242.1.

Dougherty, E. R., 1992: *An Introduction to Morphological Image Processing*. SPIE Optical Engineering Press, 161 pp.

Doviak, R. J., and D. S. Zrnić, 1993: *Doppler Radar and Weather Observations*. Dover Publications, 562 pp.

Dueben, P. D., M. G. Schultz, M. Chantry, D. J. Gagne, D. M. Hall, and A. McGovern, 2022: Challenges and benchmark datasets for machine learning in the atmospheric sciences: Definition, status, and outlook. *Artif. Intell. Earth Syst.*, **1**, e210002, https://doi.org/10.1175/AIES-D-21-0002.1.

Gadepally, V., and Coauthors, 2022: Developing a series of AI challenges for the United States Department of the Air Force. *2022 IEEE High Performance Extreme Computing Conf. (HPEC)*, Waltham, MA, Institute of Electrical and Electronics Engineers, 1–7, https://doi.org/10.1109/HPEC55821.2022.9991948.

Haupt, S. E., W. Chapman, S. V. Adams, C. Kirkwood, J. S. Hosking, N. H. Robinson, S. Lerch, and A. C. Subramanian, 2021: Towards implementing artificial intelligence post-

processing in weather and climate: Proposed actions from the Oxford 2019 workshop. *Philos. Trans. Roy. Soc.*, **A379**, 20200091, https://doi.org/10.1098/rsta.2020.0091.

Haynes, K., R. Lagerquist, M. McGraw, K. Musgrave, and I. Ebert-Uphoff, 2023: Creating and evaluating uncertainty estimates with neural networks for environmental-science applications. *Artif. Intell. Earth Syst.*, **2**, e220061, https://doi.org/10.1175/AIES-D-22-0061.1.

He, K., X. Zhang, S. Ren, and J. Sun, 2016: Deep residual learning for image recognition. *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, Las Vegas, NV, Institute of Electrical and Electronics Engineers, 770–778, https://doi.org/10.1109/CVPR.2016.90.

Hsu, W., and A. H. Murphy, 1986: The attributes diagram a geometrical framework for assessing the quality of probability forecasts. *Int. J. Forecasting*, **2**, 285–293, https://doi.org/10.1016/0169-2070(86)90048-8.

Hu, W., M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, and J. Leskovec, 2020: Open graph benchmark: Datasets for machine learning on graphs. *NIPS'20: Proceedings of the 34th International Conference on Neural Information Processing Systems*, Curran Associates, Inc., 22 118–22 133, https://dl.acm.org/doi/10.5555/3495724.3497579.

Johnson, J. M., and T. M. Khoshgoftaar, 2019: Survey on deep learning with class imbalance. *J. Big Data*, **6**, 27, https://doi.org/10.1186/s40537-019-0192-5.

Johnson, J. T., P. L. MacKeen, A. Witt, E. D. W. Mitchell, G. J. Stumpf, M. D. Eilts, and K. W. Thomas, 1998: The storm cell identification and tracking algorithm: An enhanced WSR-88D algorithm. *Wea. Forecasting*, **13**, 263–276, https://doi.org/10.1175/1520-0434(1998)013<0263:TSCIAT>2.0.CO;2.

Lagerquist, R., A. McGovern, C. R. Homeyer, D. J. Gagne II, and T. Smith, 2020: Deep learning on three-dimensional multiscale data for next-hour tornado prediction. *Mon. Wea. Rev.*, **148**, 2837–2861, https://doi.org/10.1175/MWR-D-19-0372.1.

Liu, R., J. Lehman, P. Molino, F. Petroski Such, E. Frank, A. Sergeev, and J. Yosinski, 2018: An intriguing failing of convolutional neural networks and the CoordConv solution. *NIPS'18: Proceedings of the 32nd International Conference on* Neural Information Processing Systems, Curran Associates, Inc., 9628–9639, https://dl.acm.org/doi/10.5555/3327546.3327630.

Mahalik, M. C., B. R. Smith, K. L. Elmore, D. M. Kingfield, K. L. Ortega, and T. M. Smith, 2019: Estimates of gradients in radar moments using a linear least squares derivative technique. *Wea. Forecasting*, **34**, 415–434, https://doi.org/10.1175/WAF-D-18-0095.1.

Natarajan, N., I. S. Dhillon, P. K. Ravikumar, and A. Tewari, 2013: Learning with noisy labels. *NIPS'13: Proceedings of the 27th International Conference on Neural Information Processing Systems*, Curran Associates Inc., 1196–1204, https://dl.acm.org/doi/10.5555/2999611.2999745.

NCEI, 2024: NCDC storm events database. Accessed 1 February 2024, https://www.ncdc.noaa.gov/stormevents/.

Northcutt, C. G., A. Athalye, and J. Mueller, 2021: Pervasive label errors in test sets destabilize machine learning benchmarks. *Proc. 35th Conf. on Neural Information Processing Systems Track on Datasets and Benchmarks*, Neural Information Processing Systems Foundation, 1–13, https://datasets-benchmarks-proceedings.neurips.cc/paper_files/paper/2021/file/f2217062e9a397a1dca429e7d70bc6ca-Paper-round1.pdf.

NWS, 2024: How the NWS determines wind damage and tornadoes. Accessed 3 June 2024, https://www.weather.gov/bgm/helpSurveys.

OFCM, 2021: WSR-88D Meteorological Observations, Federal Meteorological Handbook No. 11. Office of the Federal Coordinator for Meteorological Services and Supporting Research, 24 pp., https://www.icams-portal.gov/resources/ofcm/fmh/FMH11/2021_fmh11_parta.pdf.

Olson, R. S., W. L. Cava, P. Orzechowski, R. J. Urbanowicz, and J. H. Moore, 2017: PMLB: A large benchmark suite for machine learning evaluation and comparison. *BioData Min.*, **10**, 36, https://doi.org/10.1186/s13040-017-0154-4.

Pedregosa, F., and Coauthors, 2011: Scikit-learn: Machine learning in python. *J. Mach. Learn. Res.*, **12**, 2825–2830.

Potvin, C. K., C. Broyles, P. S. Skinner, and H. E. Brooks, 2022: Improving estimates of U.S. tornado frequency by accounting for unreported and underrated tornadoes. *J. Appl. Meteor. Climatol.*, **61**, 909–930, https://doi.org/10.1175/JAMC-D-21-0225.1.

Prabhat, and Coauthors, 2021: ClimateNet: An expert-labeled open dataset and deep learning architecture for enabling high-precision analyses of extreme weather. *Geosci. Model Dev.*, **14**, 107–124, https://doi.org/10.5194/gmd-14-107-2021.

Rasp, S., P. D. Dueben, S. Scher, J. A. Weyn, S. Mouatadid, and N. Thuerey, 2020: WeatherBench: A benchmark data set for data-driven weather forecasting. *J. Adv. Model. Earth Syst.*, **12**, e2020MS002203, https://doi.org/10.1029/2020MS002203.

——, and Coauthors, 2023: WeatherBench 2: A benchmark for the next generation of data-driven global weather models. arXiv, 2308.15560v2, https://doi.org/10.48550/arXiv.2308.15560.

Roebber, P. J., 2009: Visualizing multiple measures of forecast quality. *Wea. Forecasting*, **24**, 601–608, https://doi.org/10.1175/2008WAF2222159.1.

Russakovsky, O., and Coauthors, 2015: ImageNet large scale visual recognition challenge. *Int. J. Comput. Vis.*, **115**, 211–252, https://doi.org/10.1007/s11263-015-0816-y.

Ryzhkov, A. V., T. J. Schuur, D. W. Burgess, and D. Zrnić, 2005: Polarimetric tornado detection. *J. Appl. Meteor.*, **44**, 557–570, https://doi.org/10.1175/JAM2235.1.

Sandmæl, T. N., and Coauthors, 2023: The tornado probability algorithm: A probabilistic machine learning tornadic circulation detection algorithm. *Wea. Forecasting*, **38**, 445–466, https://doi.org/10.1175/WAF-D-22-0123.1.

Simonyan, K., and A. Zisserman, 2014: Very deep convolutional networks for large-scale image recognition. arXiv, 1409.1556v6, https://doi.org/10.48550/arXiv.1409.1556.

Smith, T. M., and K. L. Elmore, 2004: The use of radial velocity derivatives to diagnose rotation and divergence. *11th Conf. on Aviation, Range, and Aerospace*, Hyannis, MA, Amer. Meteor. Soc., P5.6, https://ams.confex.com/ams/pdfpapers/81827.pdf.

——, and Coauthors, 2016: Multi-radar multi-sensor (MRMS) severe weather and aviation products: Initial operating capabilities. *Bull. Amer. Meteor. Soc.*, **97**, 1617–1630, https://doi.org/10.1175/BAMS-D-14-00173.1.

Song, H., M. Kim, D. Park, Y. Shin, and J.-G. Lee, 2023: Learning from noisy labels with deep neural networks: A survey. *IEEE Trans. Neural Networks Learn. Syst.*, **34**, 8135–8153, https://doi.org/10.1109/TNNLS.2022.3152527.

Srivastava, N., G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, 2014: Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, **15**, 1929–1958.

Su, J. K., 2024: On truthing issues in supervised classification. *J. Mach. Learn. Res.*, **25** (1), 1–91.

Thiyagalingam, J., M. Shankar, G. Fox, and T. Hey, 2022: Scientific machine learning benchmarks. *Nat. Rev. Phys.*, **4**, 413–420, https://doi.org/10.1038/s42254-022-00441-7.

Tsagalidis, E., and G. Evangelidis, 2022: Exploiting domain knowledge to address class imbalance in meteorological data mining. *Appl. Sci.*, **12**, 12402, https://doi.org/10.3390/app122312402.

Vasudevan, V., B. Caine, R. Gontijo-Lopes, S. Fridovich-Keil, and R. Roelofs, 2022: When does dough become a bagel? Analyzing the remaining mistakes on ImageNet. *NIPS'22: Proceedings of the 36th International Conference on Neural Information Processing Systems*, Curran Associates Inc., 6720–6734, https://dl.acm.org/doi/abs/10.5555/3600270.3600757.

Veillette, M. S., S. Samsi, and C. J. Mattioli, 2020: SEVIR: A storm event imagery dataset for deep learning applications in radar and satellite meteorology. *NIPS'20: Proceedings of the 34th International Conference on Neural Information Processing Systems*, Curran Associates Inc., 22 009–22 019, https://dl.acm.org/doi/10.5555/3495724.3497570.

——, J. M. Kurdzo, P. M. Stepanian, J. McDonald, S. Samsi, and J. Y. N. Cho, 2023: A deep learning–based velocity dealiasing algorithm derived from the WSR-88D Open Radar Product Generator. *Artif. Intell. Earth Syst.*, **2**, e220084, https://doi.org/10.1175/AIES-D-22-0084.1.

WDTB, 2011: Distance learning operations course. Topic 5: Base and derived products. https://training.weather.gov/wdtd/courses/rac/outline.php.

Wilkinson, M. D., and Coauthors, 2016: The FAIR Guiding Principles for scientific data management and stewardship. *Sci. Data*, **3**, 160018, https://doi.org/10.1038/sdata.2016.18.

Wu, Z., B. Ramsundar, E. N. Feinberg, J. Gomes, C. Geniesse, A. S. Pappu, K. Leswing, and V. Pande, 2017: MoleculeNet: A benchmark for molecular machine learning. *Chem. Sci.*, **9**, 513–530, https://doi.org/10.1039/C7SC02664A.

Zeng, Q., Z. Qing, M. Zhu, F. Zhang, H. Wang, Y. Liu, Z. Shi, and Q. Yu, 2022: Application of random forest algorithm on tornado detection. *Remote Sens.*, **14**, 4909, https://doi.org/10.3390/rs14194909.

Zhang, C.-B., P.-T. Jiang, Q. Hou, Y. Wei, Q. Han, Z. Li, and M.-M. Cheng, 2021: Delving deep into label smoothing. *IEEE Trans. Image Process.*, **30**, 5984–5996, https://doi.org/10.1109/TIP.2021.3089942.