A Deep Learning–Based Velocity Dealiasing Algorithm Derived from the WSR-88D Open Radar Product Generator

MARK S. VEILLETTE,a JAMES M. KURDZO,a PHILLIP M. STEPANIAN,a JOSEPH MCDONALD,a SIDDHARTH SAMSI,a AND JOHN Y. N. CHOa

a Lincoln Laboratory, Massachusetts Institute of Technology, Lexington, Massachusetts

(Manuscript received 29 November 2022, in final form 6 March 2023, accepted 7 March 2023)

ABSTRACT: Radial velocity estimates provided by Doppler weather radar are critical measurements used by operational forecasters for the detection and monitoring of life-impacting storms. The sampling methods used to produce these measurements are inherently susceptible to aliasing, which produces ambiguous velocity values in regions with high winds and needs to be corrected using a velocity dealiasing algorithm (VDA). In the United States, the Weather Surveillance Radar-1988 Doppler (WSR-88D) Open Radar Product Generator (ORPG) is a processing environment that provides a world-class VDA; however, this algorithm is complex and can be difficult to port to other radar systems outside the WSR-88D network. In this work, a deep neural network (DNN) is used to emulate the two-dimensional WSR-88D ORPG dealiasing algorithm. It is shown that a DNN, specifically a customized U-Net, is highly effective for building VDAs that are accurate, fast, and portable to multiple radar types. To train the DNN model, a large dataset is generated containing aligned samples of folded and dealiased velocity pairs. This dataset contains samples collected from WSR-88D Level-II and Level-III archives and uses the ORPG dealiasing algorithm output as a source of truth. Using this dataset, a U-Net is trained to produce the number of folds at each point of a velocity image. Several performance metrics are presented using WSR-88D data. The algorithm is also applied to other non-WSR-88D radar systems to demonstrate portability to other hardware/software interfaces. A discussion of the broad applicability of this method is presented, including how other Level-III algorithms may benefit from this approach.

SIGNIFICANCE STATEMENT: Accurate and timely estimates of wind within storms are critically important for a number of applications, including severe storm nowcasting, maritime operational planning, aviation forecasting, and public safety coordination. Velocity aliasing is a common artifact that requires data quality control. While velocity dealiasing algorithms (VDAs) have been developed for decades, they remain a computationally complex and challenging problem. This paper presents an application of deep neural networks (DNNs) to increase the computational efficiency and portability of VDAs. A DNN is trained to emulate an operational algorithm, and performance is quantified over a large dataset. This work gives a convincing example of the benefits that deep learning can provide for radar algorithms, and future work highlighting these opportunities is discussed.

KEYWORDS: Weather radar signal processing; Deep learning; Neural networks; Radars/radar observations; Machine learning

1. Introduction

Meteorological radars are an integral piece of the weather enterprise’s toolset for observing and predicting weather with relatively high spatial and temporal resolution and across broad swaths of area as compared with in situ observation methods. Weather radar observations are essential for a wide range of meteorological applications including quantitative precipitation estimation (QPE; Hong and Gourley 2014), tornado detection and warning (e.g., Brown et al. 1978), identifying severe wind and hail hazards (e.g., Burgess and Lemon 1990; Ortega et al. 2016), characterizing tornado/mesoscale dynamics (e.g., Wurman et al. 1996), forecasting real-time mesoscale snowbands (e.g., Kristovich et al. 2003), and hurricane rainbands (e.g., Biggerstaff et al. 2021), and assimilation into numerical weather prediction systems (e.g., Stensrud et al. 2009).

To enable these applications, the U.S. National Weather Service (NWS) operates a network of Weather Surveillance Radar-1988 Doppler (WSR-88D) meteorological radar sets that provide real-time atmospheric observations over the United States and outlying territories. The ability of WSR-88D estimates to successfully support the NWS mission is contingent on a series of processing steps that convert raw in-phase and quadrature receiver base-band samples (I/Q; termed “Level-I” data) into a collection of standard radar data. The most basic of these data for a dual-polarization radar are the six “Level-II” variables that are calculated through time series processing of the Level-I signals: horizontal reflectivity factor, radial velocity, Doppler spectrum width, differential reflectivity, copolar correlation coefficient, and differential phase. These Level-II variables contain information relating to the size, shape, abundance, composition, and motion of objects in the atmosphere. A number of additional “Level-III” products are generated within the WSR-88D Open Radar Product Generator (ORPG) using more complex workflows and often incorporate two-, three-, or four-dimensional analyses across multiple Level-II variables, sometimes in combination with
ancillary datasets beyond just radar (e.g., model data). For example, the ORPG QPE algorithm combines multiple Level-II variables and integrates estimated rainfall over time in a Level-III product. As a result of these sophisticated processing techniques, Level-III products often have a greater degree of quality control, are more directly relatable to common users of radar data, and provide more intuitive visual interpretation of the phenomena that they are meant to characterize, making them well suited for use by weather forecasters and the general public.

One of the major challenges in weather radar is data quality control. In particular, range-Doppler ambiguities, a direct result of the Doppler dilemma (Doviak and Zrnić 1993; Skolnik 2002), are a significant challenge for Doppler weather radars. The main focus of this work is velocity aliasing, which causes ambiguous velocity values to be estimated in regions where winds exceed the Nyquist velocity $V_n$. The result of this ambiguity is that the magnitude of the velocity vector—the wind speed relative to the radar—can be any of several different values, and even the sign of the radial wind direction—either toward or away from the radar—is unknown. Applications relying on accurate wind measurements require eliminating these ambiguities, revealing the singular “true” radial wind velocity value. Any winds that exceed a nominal maximum (max) unambiguous velocity are subject to aliasing. This maximum unambiguous velocity is a function of the radar wavelength and the pulse repetition frequency [PRF; see Eq. (1)].

Too high a PRF results in unambiguous ranges that are too short for operational use. At higher frequencies, such as C-band and X-band radars, the Doppler dilemma becomes even more challenging because of the shorter wavelengths. This relation is given by (Doviak and Zrnić 1993) as

$$V_n = \frac{\lambda \text{PRF}}{4},$$  \hfill (1)

where the Nyquist velocity $V_n$ is in meters per second, $\lambda$ is the wavelength in meters, and PRF is the pulse repetition frequency per second. An example of velocity aliasing is shown in Fig. 1, where a plan position indicator (PPI) of radial velocity in the left panel alternates between inbounds (negative values/blue colors) and outbounds (positive values/red colors) when traversing along the green circle. In the right panel, the aliasing effect of the velocities is observed as discontinuities that jump across the Nyquist intervals. Given the trade-off of the PRF selection, velocity dealiasing algorithms (VDAs) are important for any meteorological radar. Velocity dealiasing is particularly important in cases of severe weather and hurricanes, where radial velocities are high. In tornadoes, for example, Doppler velocities can easily exceed the Nyquist velocity by several times (e.g., Burgess et al. 2002). Level-III algorithms, such as the tornadic vortex signature (Brown et al. 1978; Mitchell et al. 1998) and mesocyclone detection algorithm (Stumpf et al. 1998), depend on accurate velocity dealiasing.

While the WSR-88D VDA demonstrates exceptional performance and ongoing improvements within each ORPG build, the algorithm is relatively complex and requires close to 1 s per tilt (constant-elevation-angle 360° azimuthal scan) for complex storm environments. While this is acceptable on appropriate hardware and for the current scan rates for the WSR-88D (which peak at 4.8 revolutions per minute and must keep up in “real time”), there are several reasons why a simpler/faster implementation would be highly desirable. For example, future radar systems, such as phased arrays, may scan much faster (Weber et al. 2021), resulting in the need for algorithms that are capable of running more efficiently. Several ORPG algorithms take much more than 1 s to run, making more efficient performance for future faster-scanning radars a must. In addition, the ORPG VDA utilizes environmental wind information, which is not always accessible in real time or when processing large batches of archived data. An accurate, fast, and highly portable algorithm that only requires a single velocity image would be of great value in situations like these.
For these reasons, we utilize machine learning (ML), specifically deep learning (DL), to emulate the two-dimensional WSR-88D VDA using purely data-driven approaches. DL approaches have been successfully demonstrated in diverse meteorological applications such as precipitation nowcasting (Ravuri et al. 2021; Xingjian et al. 2015), subseasonal forecasting (Weyn et al. 2021), satellite precipitation estimation, (Wang et al. 2021), synthetic weather radar (Veillette et al. 2018; Hilburn et al. 2021), mesocyclone classification (Krinitskiy et al. 2018), extreme weather event detection (Racah et al. 2017), and several others. Within radar meteorology specifically, a range of different ML algorithms has been used, and many algorithms were summarized by Chandrasekar (2020), with focuses on QPE, precipitation classification, and nowcasting. For example, Chen et al. (2020) use deep neural networks to estimate precipitation using both satellite-borne and ground-based radar networks. Other QPE-based DL approaches include those by Yo et al. (2021), Husnoo et al. (2021) (using the U.K. weather radar network), and Peng et al. (2022). Although not DL-centric, precipitation classification has used other forms of ML, including fuzzy logic (Park et al. 2009) and Bayesian approaches (Yang et al. 2019).

Currently, only one operational ORPG algorithm utilizes modern machine learning techniques (the chaff detection algorithm; Kurdzio et al. 2017a), but, because of the wide accessibility to years of archived data, the resources exist to enable other ML-based algorithms (e.g., Jatau et al. 2021). A DL-based VDA should enable a faster algorithm, provide a WSR-88D-quality dealiaser for users of other radar datasets, and possibly be implementable within open-source software packages. The VDA is an ideal algorithm to attempt to replicate within the ORPG, because it uses just one moment (the velocity field), is computationally complex (with a desire to decrease resources, especially for non-ORPG hardware), is a critical need for non-WSR-88D instruments, and performs well enough to use as an unsupervised truthed dataset. In this work, we specifically attempt to emulate the WSR-88D velocity dealiasing algorithm. This is an important distinction, because we are not expecting to necessarily improve upon the algorithm. The reason for this approach is that the existing algorithm can serve as an exceptionally large truthed dataset. The Level-III velocity product can be considered truth at all times if we are attempting to emulate/recreate the algorithm, eliminating the need for any manual dealiasing.

While improving upon the algorithm would be ideal, this approach sets the stage for future recreation of other ORPG algorithms using ML techniques for both computational efficiency and portability. We explore different techniques to optimize the algorithm, approaches for improving upon baseline results, and applications beyond the WSR-88D to other radars. The paper is organized as follows: section 2 discusses prior work in the VDA space; section 3 describes the data and the approach; section 4 discusses the chosen model architecture; section 5 presents results; and section 6 provides a discussion of the applicability of the results, along with some conclusions.

2. Prior work

Since some of the earliest work on velocity ambiguities by Ray and Ziegler (1977), several approaches to dealiasing constant-PRF radar radial velocity data have been presented in the literature, and a subset of these has been deployed operationally in the ORPG. The initial ORPG VDA was based on Elts and Smith (1990); it takes a radial-by-radial approach to unfolding velocities. Each radial is compared with the previous radial using a local environment dealiasing (LED) technique. These radial continuity checks are based only in the radial dimension, making the algorithm computationally efficient. As discussed in Witt et al. (2009), as well as Zittel et al. (2011), the Elts and Smith (1990) approach worked well in an initial ORPG implementation with limited computational resources but often failed in cases of lower Nyquists (i.e., volume coverage pattern 31) and strong shear (mesocyclones, gust fronts, etc.). As new computational capabilities were added, it became possible to implement a new ORPG VDA based on the work in Jing and Wiener (1993), which takes a two-dimensional approach to solving a linear system of equations in a connected two-dimensional region by minimizing discontinuities in a least squares problem. The ORPG version of the algorithm adds a weighting factor to Eq. (5) of Jing and Wiener (1993), which aids with issues in noisy data (Witt et al. 2009). A global dealiaser using environmental data is also applied as part of Jing and Wiener (1993) and is used in the ORPG VDA as the second step of a “two-phased dealiasing approach” (Witt et al. 2009). Anecdotally, the ORPG VDA performs admirably in most situations; however, the two-dimensional VDA is still relatively complex and computationally expensive (Witt et al. 2009; Zittel et al. 2011; NWS2021).

The current WSR-88D VDA is available via the ORPG Common Operations and Development Environment (CODE) software package (CODE 2021), an open-source version of the ORPG. As of this writing, the CODE ORPG Build 20 source code (written mainly in the C programming language) is available to the public (https://www.weather.gov/code88d/). To the authors’ knowledge, there is no widespread use of the WSR-88D VDA in other operational radars or research radars. Beyond 2D VDAs (the main focus of this work), other approaches that utilize additional dimensions have been developed. For example, James and Houze (2001) present a real-time, four-dimensional dealiasing (4DD) scheme that takes a top-tot-down approach using the elevation and time dimensions in addition to investigating neighboring gates and radials. This technique is implemented along with other dealiasing algorithms in the Python ARM Radar Toolkit (Py-ART) (Helmus and Collis 2016). These algorithms are also often similarly (or more) computationally complex in comparison with the WSR-88D algorithm, frequently taking a nontrivial amount of time to run, especially in complex velocity field situations. Additionally, at C-band and higher frequencies, with low Nyquist values, automated dealiasers perform particularly poorly, especially in tornados (e.g., Bluestein et al. 2007; Yu et al. 2007). For this reason, most research radars (e.g., mobile C-band, X-band, Ka-band, and W-band radars) that collect tornado data must have their velocities manually dealised, step-by-step in time and elevation (e.g., Oye et al. 1995). This onerous task is extremely time consuming and often unavoidable. Even operational radars at C band, especially those in areas of sparse data and high environmental shear, can suffer
from dealiaser failures, such as the Australian weather radar network (Louf et al. 2020).

After the 4DD approach was developed, additional dimensions were explored for dealiasing, especially with low-Nyquist radar systems. For example, Zhang and Wang (2006) discuss multipass filters that determine confidence levels that subsequently increase with additional passes of dealiasing. Similar to other multidimensional approaches, multiple passes add time and complexity to the processing steps despite performing exceptionally well. Xu et al. (2011) used VAD profiles to dealias radar data for numerical weather prediction assimilation. While their methods perform well in many case examples, they tend to break down in areas of localized high shear (mesocyclones, tornadoes, hurricane eyewalls, etc.). He et al. (2012) developed a C-band-specific approach to dealiasing for the Chinese CINRAD radar network, which operates at a single PRF with a high frequency, making them particularly susceptible to velocity aliasing issues.

More recently, the Unfold Radar Velocity (UNRAVEL) algorithm takes a modular approach to a VDA, allowing for an optimization of results that can balance between performance needs and computational complexity (Louf et al. 2020). UNRAVEL currently contains two different (but expandable) dealiasing strategies: a tilt-by-tilt (two dimensional) and a three-dimensional continuity check. Louf et al. (2020) use artificially aliased S-band data as a reference for analyzing performance at C band, a technique that is adopted in a similar fashion in this study. Finally, Feldmann et al. (2020) introduce the region-based recursive Doppler dealiasing method (R2D2) for dealiasing in highly sheared environments, a historically difficult area for VDAs. A four-dimensional procedure with a two-dimensional continuity check iterates until improvement is no longer achieved, and the highly sheared areas are given a “focus” with an image mask and a spatial buffer, resulting in exceptional performance.

3. Data description

To establish notation, we will represent the true radial velocity $V$ at a gate as the sum of the observed velocity that has potentially been subject to aliasing, $V_{\text{obs}}$, and an even multiple of the Nyquist velocity $V_{\text{Ny}}$, that is,

$$V = V_{\text{obs}} + 2nV_{\text{Ny}},$$

where $n \in \mathbb{Z}$ represents the number of “folds” required to dealias the observed folded velocity. We represent a single tilt of $V_{\text{obs}}$ and $V$ as a two-dimensional array of shape $N_{\text{az}} \times N_{\text{rang}}$ representing the azimuthal and radial dimensions, respectively. The goal of a VDA is to infer the correct value of $n$ at each gate so that the true velocity can be reconstructed using Eq. (2). In this work, this task is accomplished by learning a deep neural network (DNN) $n = f(V_{\text{obs}}; \theta)$ that outputs the correct fold number for all gates using a single tilt of $V_{\text{obs}}$. The variable $\theta$ in this function denotes weights of a DNN that will be learned during training. Note that a two-dimensional VDA was chosen to maximize portability of the algorithm by not requiring additional inputs such as environmental winds. However, the methods described below can be extended to higher-dimensional versions of the problem (e.g., 4DD) that incorporate additional tilts, time steps, radar variables, environmental winds, and/or other inputs if desired. This is an important aspect of the potential for DL to vastly improve the run times of these more complex algorithms.

This proposed VDA is entirely data driven and requires a large number of aligned samples containing input aliased velocity, associated Nyquist velocities, and dealiased target velocity: $(V_{\text{obs}}, V_{\text{Ny}}, V)$. Below, we describe the sources used to build datasets for the development and evaluation of a DNN-based VDA. In addition, we describe the processing pipeline implemented to automatically construct labeled training samples required for DNN training and quantitative assessments.

a. Training data description

Training data for dealiasing were derived entirely from WSR-88D data products. As described in the introduction, WSR-88D data are available from two tiers of products, Level-II variables and Level-III products that are derived from Level-II variables. Of particular interest for this work is the Level-III radial velocity product, which is a dealiased and data quality edited version of the (potentially aliased) Level-II velocity.

While no perfect “truth” exists for this task (outside manually labeled data), Level-III velocity was chosen as truth for a DL implementation because it offers a high-quality, operationally hardened reference and is anecdotally seen as a gold standard among forecasters. Archives of both Level-II and Level-III products can be downloaded for free from various cloud providers—for example, Google Cloud Storage (https://cloud.google.com/storage/docs/public-datasets/nexrad).

1) PREPROCESSING AND LABELING

Training data requires that pairs of input velocity $V_{\text{obs}}$ and target dealiased velocity $V$ be coaligned gate by gate. For this reason, in an approach similar to Louf et al. (2020), training targets were created by artificially aliasing Level-III data using an input Nyquist velocity $V_{\text{Ny}}$ assumed in this work to be constant over each two-dimensional radar tilt.

The processing pipeline for creating a training pair is described as follows: given a WSR-88D site ID and a time stamp, base Level-III velocity data (code N0U within the ORPG used by the Radar Operations Center) are downloaded and extracted into a two-dimensional array with a height dimension of $N_{\text{az}} = 360$ azimuths and width dimension truncated to the first $N_{\text{rang}} = 1152$ radial gates (this choice of radial dimension made training more convenient but is not required during testing). The azimuthal dimension of each array is sorted such that the radial with minimum azimuth is located at the top of the array (index 0), ensuring a consistent azimuthal indexing across the training data. Since the Level-III velocity algorithm clips data to the range of $[-63.5, 63.5 \text{ m s}^{-1}]$, regions in which the end points of this clipping range are observed need to be filled in using data from the nearest Level-II velocity field, which is downsampled by averaging radial in each 1° arc. Note that the dealiasing of Level-II data for filling data
gaps is possible since it is known that the magnitude of velocity within clipped regions exceeds the limits of Level-III data, and hence, within clipped regions, the Nyquist velocity can be added/subtracted until values in the region exceed the clipped value.

After extracting, reindexing, and backfilling clipped regions, the resulting two-dimensional \( V \) arrays and associated Nyquist velocities are saved to disk using Tensorflow Record format (https://www.tensorflow.org/tutorials/load_data/tfrecord), which is a simple and efficient format for training DNNs using large datasets. At train time, a data loader reads a batch of two-dimensional dealiased velocity grids \( V \) and Nyquist velocities \( V_n \) and constructs a batch of artificially folded velocities \( V_a \) along with label arrays \( f \) containing the target fold numbers \( n \). To increase the difficulty of dealiasing, the Nyquist velocities used operationally were lowered by an amount of 10 m s\(^{-1}\) (a hyperparameter) so that the DNN would be exposed to more folded regions. This was also done to increase the likelihood that the DNN will generalize to radars that commonly use lower Nyquist velocities than does the S-band WSR-88D (such as radars at higher frequencies).

The label array \( f \) used for training is of shape \( N_{az} \times N_{rng} \times 6 \), where the last dimension is a “one-hot” encoding representing the categories shown in Table 1. Note that fold values with \(|n| > 2\) are not included as a separate category as they are exceptionally rare in WSR-88D data. This makes use of the algorithm in some situations and with certain radar types less tenable, such as higher-frequency mobile radars in cases of tornados (e.g., Kurdzo et al. 2017b) and hurricanes (e.g., Alford et al. 2019). However, some examples of successful unfolding with these types of radars are shown in section 6. Using the fold array \( f \), the artificially folded array \( V_a \) was constructed by subtracting or adding the appropriate number of Nyquist velocities. The data were padded by 12° in the azimuthal dimension with a periodic boundary condition to create arrays with shape \( N_{az} = 384 \). An example of this processing can be seen in Fig. 2, where the original Level-III velocity is shown on the left, with regions exceeding the Nyquist velocity contoured in black. The artificially aliased velocity is shown in the center panel, and the resulting fold number \( n \) is shown on the right.

The result of this pipeline is four variables which are grouped as inputs: \( X = (V_a, V_n) \) and outputs: \( Y = (F, V) \). These samples are grouped into minibatches and used for DNN training.

2) EVENT SELECTION

Selecting a training dataset that includes sufficient representation of strong radial velocities and highly sheared environments is critical for the success of the algorithm. Storm events that contain strong and dynamic winds, such as microbursts, mesocyclones, tornados, and hurricanes, are not only more difficult to dealias due to sudden changes in wind magnitude and direction, but also represent the greatest impact operationally. Since these events are statistically rare, random sampling of events will likely underrepresent them. To ensure the training and testing datasets provide adequate coverage of these events, the NCEI Storm Event Database (https://www.ncdc.noaa.gov/stormevents/) was used to identify radar sites and times where these high-impact storms were reported. In addition to events with confirmed severe storm occurrences, a set of “null events” was also selected where either no storms were reported or warnings were issued and no confirmed cases were surveyed.

In all, the resulting dataset derived from WSR-88D was selected from 6350 separate events from the Storm Event Database that fall between January 2017 and June 2022 (66 months). For each of these events, several velocity images were sampled, totaling 427 491 processed velocity images of shape \( 360 \times 1152 \). This dataset was split into three subsets: 80% for training sampled between January 2017 and October 2020, 8% for

<table>
<thead>
<tr>
<th>Range</th>
<th>Fold no.</th>
<th>One-hot encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td>Background values</td>
<td>—</td>
<td>(1, 0, 0, 0, 0, 0)</td>
</tr>
<tr>
<td>( V \leq -3V_n )</td>
<td>( n = -2 )</td>
<td>(0, 1, 0, 0, 0, 0)</td>
</tr>
<tr>
<td>( -3V_n &lt; V \leq -V_n )</td>
<td>( n = -1 )</td>
<td>(0, 0, 1, 0, 0, 0)</td>
</tr>
<tr>
<td>( -V_n &lt; V \leq V_n )</td>
<td>( n = 0 )</td>
<td>(0, 0, 0, 1, 0, 0)</td>
</tr>
<tr>
<td>( V_n &lt; V \leq 3V_n )</td>
<td>( n = +1 )</td>
<td>(0, 0, 0, 0, 1, 0)</td>
</tr>
<tr>
<td>( V &gt; 3V_n )</td>
<td>( n = +2 )</td>
<td>(0, 0, 0, 0, 0, 1)</td>
</tr>
</tbody>
</table>

Fig. 2. Technique for creating targets for training: (left) Regions of Level-III velocity that exceed the Nyquist velocity are identified (black contours). (center) These regions are aliased so that data fall in the range \([-V_n, +V_n]\). (right) Regions that are aliased by a factor of \( f \) are stored in a label mask that is used as the final training target.
validation sampled between November 2020 and May 2021, and 12% for testing sampled between June 2021 and June 2022. Partitioning the data by time was done to avoid potential overlaps in storm events and/or radars. When saved to disk and compressed, this combined dataset totaled 229 gigabytes in size.

b. Testing data description

In addition to the processed Level-III test data mentioned in the previous section, Level-II velocity as well as velocity data from other radars were also used for testing. The other radars included for verification purposes are the PX-1000 X-band radar (Cheong et al. 2013) and the C-band Terminal Doppler Weather Radar (TDWR; Michelson et al. 1990; Cho and Weber 2010). PX-1000 is a transportable, solid-state, polarimetric radar that has been used to study severe weather, tornadoes, and rainfall, as well as serve as a system for radar engineering, teaching, and research. The TDWR is an operational radar that is part of a network of 45 systems across the United States, and it serves as a primary tool for detecting wind shear and hazardous convection at large airports (Evans and Turnbull 1989).

The characteristics of the different test sets are described in Table 2. One challenge in using non-Level-III data is that there is no ideal dealiased reference to use for comparison, so verification for non-Level-III-derived velocity will be mostly qualitative and will utilize multiple views of the same event to increase confidence that the DNN VDA is correctly identifying the number of folds in the observed data. Quantitative scoring will be performed only in the case of artificially aliased Level-III data, where a truth is available.

### 4. Model description

This section describes the architecture of the DNN model \( f(\cdot; \theta) \). Dealiasing is framed here as an image-segmentation task, which amounts to classifying each pixel in a velocity image as one of 6 categories corresponding to fold numbers \( n = -2, -1, 0, 1, 2, \) or background. After the most likely fold number is identified for each pixel, the input aliased velocity is modified by the correct scaling of Nyquist velocity using Eq. (2).

The particular DNN model chosen for this task is a U-Net (Ronneberger et al. 2015), which is a popular neural network architecture that has been applied in a number of areas, including image segmentation (Iglovikov and Shvets 2018), biomedical image analysis (Haque and Neubert 2020), and image-to-image translation (Isola et al. 2017). The U-Net has also been widely adopted in meteorological applications, including nowcasting (Agrawal et al. 2019), precipitation estimation (Wang et al. 2021), satellite image gap filling (Xing et al. 2022), and several others. What makes the U-Net an effective model is its ability to learn and process features at multiple spatial scales across an image. This property makes the architecture ideal for dealiasing as it requires identifying sudden changes in velocities at fine scales and identifying wind patterns across the whole image without the need for environmental wind information, in order to determine the correct fold numbers.

Another beneficial property of the U-Net architecture is that it is fully convolutional, making it possible to perform inference on images that are of a different radial shape \( N_{\text{rng}} \) when compared with the training data. Different azimuthal shapes in testing data can also be used; however, the resolution should be in \( 1^\circ \) steps, and it is recommended that inputs contain a full \( 360^\circ + 12^\circ \) of periodic boundary conditions \( (N_{\text{az}} = 384) \). The portability of the U-Net architecture will be demonstrated in a later section by applying the trained model to velocity data of different shapes. It is important to note that cases with sector scans, as can be common in mobile radars, are not yet technically supported with this technique. However, modifications can be made to the data and the algorithm in the future, including zero-padding of azimuths that are missing. While this has not been specifically tested, it would be a straightforward approach to using the U-Net on these types of datasets. A more difficult problem to solve would be the limitation on \( 1^\circ \) azimuthal steps, which, as of the writing of this study, will limit the algorithm’s utility in some research settings. Future work, including input from the research community, will include these types of adjustments.

The details of the U-Net architecture used for dealiasing are shown in Fig. 3. The network takes two inputs. The first input is a batch of observed (aliased) velocity \( V_{\text{a}} \) with shape \([B, N_{\text{az}}, N_{\text{rng}}, 1]\), where \( B \) denotes the batch size, \( N_{\text{az}} \) is the (padded) azimuthal dimension, \( N_{\text{rng}} \) is the radial dimension, which may be of arbitrary size, and the final 1 denotes the singleton channel dimension of the input. The second input to the model is the array of Nyquist velocities for each element of the batch used to artificially fold the target images, which has shape \([B, 1]\). The aliased velocity is initially scaled by \( 1/V_{\text{a}} \), which produces values in the interval \([-1, 1]\). After normalization, background values in the original velocity are set to a placeholder value of \(-3\).

After normalization, the downward portion of the U-Net, made up of blocks labeled \( D_1, \ldots, D_4 \), extracts features from the input and maps the data to lower and lower resolutions by cutting the sizes of each dimension by factors of 2 or 4 at each stage. Each \( D \)-block is made up of a series of standard deep learning operations summarized in the top panel of Fig. 4.

<table>
<thead>
<tr>
<th>Velocity source</th>
<th>Frequency band</th>
<th>Max range (km)</th>
<th>Range resolution (km)</th>
<th>Azimuthal resolution (°)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NEXRAD Level III</td>
<td>S-Band</td>
<td>230</td>
<td>0.25</td>
<td>1</td>
</tr>
<tr>
<td>NEXRAD Level II</td>
<td>S-Band</td>
<td>230</td>
<td>0.25</td>
<td>0.5</td>
</tr>
<tr>
<td>TDWR</td>
<td>C-Band</td>
<td>90</td>
<td>0.15</td>
<td>1</td>
</tr>
<tr>
<td>PX1000</td>
<td>X-Band</td>
<td>60</td>
<td>0.03</td>
<td>1</td>
</tr>
</tbody>
</table>
The $D$-block is composed of two paths, the first being 2 two-dimensional convolutional layers with kernel size $k$ with a leaky rectified linear unit (ReLU) activation function that uses $\alpha = 0.1$. The output of the second convolution is downsampled with a $d_1$, $d_2$ average pooling layer. The number of channels (features) output by each $D$-block doubles after the initial block, and the channel dimension of the first output block $D_1$ is a configurable parameter called the starting neurons or $s_n$. The values of $k$, $d_1$, and $d_2$ for the four $D$-blocks are shown in the left portion of Table 3.

At the base of the dealiasing U-Net, low-resolution feature maps are processed by three parallel convolutional layers with kernel sizes of $k = 3, 5,$ and 7. These additional convolutions were added to the standard U-Net architecture to maximize the receptive field of the convolutions, which allows nonlocal features to be incorporated into the resulting segmentation. Larger receptive fields should help correctly identify predominate winds without the need for a reference velocity field. Experimentation with different network architectures showed that U-Nets with only a single 3 $\times$ 3 convolutional layer at the base produced validation losses that were approximately 2 times as large as architectures that use the parallel convolutions.

The right portion of the U-Net is made up of four upsampling blocks, $U_i$, $i = 1, \ldots, 4$. These blocks take two inputs. The first input is the output of each parallel $D$-block, and the second is the output of the preceding $U$-block. The steps performed in the $U$-block are shown in the bottom panel of Fig. 4. The two inputs are concatenated along the channel dimension and followed by two applications of batch normalization (Ioffe and Szegedy 2015), ReLU, upsampling, and two-dimensional convolution. In contrast to the $D$-blocks, the output of each $U$-block halves the number of features. The values of $k$, $d_1$, and $d_2$ for the four $U$-blocks are shown in the right portion of Table 3.

The output of the final $U$-block is passed to a 3 $\times$ 3 convolutional layer with linear activation to produce the one-hot encoded array $F$ with spatial dimensions equal to the input velocity. The output array $F$ has six output channels that can be interpreted as log odds of the six potential categories of each pixel. The correct fold number is computed using the argmax along the channel dimension of this array. It is
worth pointing out that this output can also be combined with a softmax activation to create probabilistic predictions across different fold numbers; however, this idea was not explored further.

5. Results

a. Model training

The model described in section 4 was implemented using the Tensorflow Keras DL framework (Bisong 2019) and trained on two NVIDIA Volta V100 graphical processing units (GPUs) located within the MIT Lincoln Laboratory TX-Green Super Computing Center (Kepner et al. 2017). Multiple hyper-parameter settings of the model were tested during development, and results are provided for a U-Net with starting neuron choices \( s_n = 8, 16, \) and 24. A categorical cross-entropy loss function was used with an Adam (Kingma and Ba 2014) optimizer for weight updates. Computations utilized mixed-precision training (Samsi et al. 2020), where network operations use a 16-bit floating point, and weight updates were performed using a 32-bit floating point. An initial learning rate of 0.0025 was used and was decreased by a factor of 0.99 every epoch. Training iterations used a data-distributed model with a batch size of 32 input images spread across the two GPUs. The model was allowed to train for 200 epochs with 600 batches used per epoch. The total time for training averaged approximately 25 h per model.

Figure 5 shows the evolution of two metrics during training for a model trained with \( s_n = 16 \). The left panel shows categorical cross-entropy loss, which describes how well the model classifies fold number across each image, and the right panel shows velocity RMSE computed between the true velocity and dealiased velocity. The training curve shows performance on the training set, which includes an artificially lowered Nyquist velocity. To ensure accuracy of the target dataset, the validation curve reflects Nyquist velocities that were used operationally for the WSR-88D, which explains why the validation results are slightly better than training curves. As can be seen by the validation loss curves, the model seems to fully converge by around epoch 125. The training epoch with the lowest-velocity RMSE on the validation set was selected for further testing.

b. Test set results

The trained U-Net with \( s_n = 16 \) was tested on several instances of the 0.5° elevation Level-II velocity data with four sample cases visualized in Fig. 6. Note that these tilts of the Level-II velocity have a 0.5° azimuthal resolution, which is half that of the training data. Since the U-Net was trained on 1° data, a single scan of Level-II \( V \) was split into a batch of two separate 1° scans by splitting every other radial. After both of these splits were dealiased using the U-Net, the result was recombined into a 0.5° dataset.

These results demonstrate the ability of the U-Net to handle challenging dealiasing situations that involve global large-scale features, as well as resolve finer-scale aliasing in situations of high shear (e.g., severe storms). The four cases shown include zoomed-out (case a) and zoomed-in (case b) views of a tornado near KGWX (Greenwood Springs, Mississippi) at 0412 UTC 14 April 2019, Hurricane Ida as it moved past KLIX (Slidell, Louisiana) at 1639 UTC 29 August 2021 (case c), and a widespread area of convection with several isolated cells near KPAH (West Paducah, Kentucky) at 0352 UTC 11 December 2021 (case d). The highly sheared tornadic signature in case b, as well as the eyewall during Hurricane Ida in case c, shows examples of how the U-Net can correctly dealias areas with rapidly shifting, strong winds in complex environments. Cases a and d show examples of how the U-Net can identify prevailing winds across an entire PPI in order to dealias isolated cells where the correct direction is not necessarily obvious without external environmental data. In each case shown, the U-Net

<table>
<thead>
<tr>
<th>D-block</th>
<th>( k )</th>
<th>( d_1 )</th>
<th>( d_2 )</th>
<th>U-block</th>
<th>( k )</th>
<th>( d_1 )</th>
<th>( d_2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( D_1 )</td>
<td>7</td>
<td>1</td>
<td>2</td>
<td>( U_1 )</td>
<td>3</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>( D_2 )</td>
<td>5</td>
<td>2</td>
<td>4</td>
<td>( U_2 )</td>
<td>3</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>( D_3 )</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>( U_3 )</td>
<td>3</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>( D_4 )</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>( U_4 )</td>
<td>3</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 3. Parameter settings for the U-Net blocks described in Fig. 4.

![Figure 5. Values of (left) the categorical cross-entropy loss and (right) RMSE of resulting dealiased velocity for the training and validation set during training. The training set uses Nyquist velocities set 10 m s\(^{-1}\) lower than the operational values for each sample, whereas the validation set used the actual Nyquist velocity, which explains why the validation set shows better performance.](image-url)
results are consistent with the Level-III reference image shown in the right-most column.

For a more quantitative assessment over a larger sample size, the U-Net VDA was scored using an artificially aliased Level-III testing dataset. In addition to loss value and average RMSE between corrected and target velocities, additional metrics of probability of detection (POD), false alarm rate (FAR), and critical success index (CSI) were also computed. To define these metrics, a hit \( H \) was defined as gates where the predicted fold number \( n_{\text{pred}} \) was nonzero and matched the actual fold number \( n_{\text{true}} \). Misses \( M \) were counted for gates where \( n_{\text{true}} \neq 0 \) and \( n_{\text{pred}} = 0 \), and false alarms \( F \) were counted for gates where \( n_{\text{pred}} \neq 0 \) and \( n_{\text{true}} = 0 \). Using these counts, POD, FAR, and CSI were computed as

![Figure 6](image-url)
Table 4. Performance of the U-Net VDA for different network sizes (controlled by the hyperparameter $s_n$) on the Level-III test set. Results from the region-based VDA implemented in the Py-ART library are also shown as a baseline. The U-Net VDA scores high across all metrics and shows small increases as the number of parameters in the model increases. The run times of different VDAs are shown, illustrating the computational speed of the DL approach in comparison with the baseline. For each metric, the arrows indicate the direction of increasing performance and the boldface values indicate the best performance.

<table>
<thead>
<tr>
<th>VDA</th>
<th>RMSE↓</th>
<th>POD↑</th>
<th>FAR↓</th>
<th>CSI↑</th>
<th>CPU Runtime (ms per scan)↓</th>
<th>GPU Runtime (ms per scan)↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>U-Net ($s_n = 8$)</td>
<td>1.6387</td>
<td>0.9827</td>
<td>0.0105</td>
<td>0.9726</td>
<td>70.6</td>
<td>16.4</td>
</tr>
<tr>
<td>U-Net ($s_n = 16$)</td>
<td>1.5300</td>
<td>0.9878</td>
<td>0.0117</td>
<td>0.9764</td>
<td>129.3</td>
<td>17.4</td>
</tr>
<tr>
<td>U-Net ($s_n = 24$)</td>
<td>1.4753</td>
<td>0.9887</td>
<td>0.0109</td>
<td>0.9781</td>
<td>206.3</td>
<td>24.0</td>
</tr>
<tr>
<td>Region-based (Py-ART)</td>
<td>5.6351</td>
<td>0.9045</td>
<td>0.1887</td>
<td>0.7472</td>
<td>1669</td>
<td>–</td>
</tr>
</tbody>
</table>

\[
\text{POD} = \frac{\text{No. } \text{H}}{\text{No. } \text{H} + \text{No. } \text{M}}, \\
\text{FAR} = \frac{\text{No. } \text{F}}{\text{No. } \text{H} + \text{No. } \text{F}}, \quad \text{and} \\
\text{CSI} = \frac{\text{No. } \text{H}}{\text{No. } \text{H} + \text{No. } \text{M} + \text{No. } \text{F}}. 
\]

To serve as an additional baseline, the region-based VDA that is implemented within Py-ART was also applied to the artificially aliased Level-III test data and scored. According to the Py-ART documentation (https://arm-doe.github.io/pyart/API/generated/pyart.correct.dealias_region_based.html), the region-based VDA works by “finding regions of similar velocities and unfolding and merging pairs of regions until all regions are unfolded. Unfolding and merging regions is accomplished by modeling the problem as a dynamic network reduction.” Similar to the U-Net, the region-based VDA is also a two-dimensional algorithm that only requires a single tilt of velocity, which is why it was selected for comparison. Also, environmental wind information was not supplied to the region-based algorithm, since the U-Net model does not leverage these additional data. The default settings of the region-based algorithm were used for this study.

The comparison of performance of the U-Net and region-based VDAs on the Level-III test set is shown in Table 4. Across all metrics, the U-Net performance exceeds that of the baseline considered, with the best performance being observed for the U-Net with the most learnable parameters ($s_n = 24$); however, the differences in performance metrics between the three U-Nets considered is small. When compared with the region-based baseline, the U-Net improves the velocity RMSE by over 300% and provides an approximately 30% increase in CSI. A further benefit of the U-Net is its computational efficiency, running 1200% faster than the region-based VDA on equivalent CPU, and an impressive 9800% faster on GPU (which is not an option for the region-based VDA). Also of note is the trade-off between skill and computation time for different $s_n$. Larger models also require additional memory to run, which may limit the ability to use these models on certain hardware.

Note that comparisons of skill scores between the U-Net and region-based VDAs reported in Table 4 should be interpreted with some care, as the U-Net has the advantage of being trained to emulate the same algorithm used to create the test set. Since the ORPG VDA utilizes environmental wind data, it is reasonable to assume that its output will be more accurate, on average, than the region-based baseline that was not supplied this additional information. However, it is possible that the U-Net learned and reproduced the same mistakes as the ORPG, leading to overly optimistic skill scores in this case. To explore this possibility further, several case studies where the ORPG and region-based VDAs disagreed were examined by human experts to judge which are correct. In virtually all cases that were examined, results similar to the example shown in Fig. 7 were observed. Three consecutive frames of Level-II velocity (left column) are dealiased by the U-Net and region-based algorithms (second and third columns). In the second row, without providing environmental wind information, the region-based algorithm produces incorrect results for isolated regions in the north and northeast portions of the scan, which disagrees with both the U-Net and Level-III reference (right column), which provide consistent results across all frames.

**CASE STUDIES**

This section considers two case studies composed of tornadoic storms that were simultaneously sampled by multiple radar systems. As such, each radar provides a unique perspective of the storm velocity structures, affected by its unique system’s aliasing characteristics. Application of U-Net VDA with $s_n = 16$ in these cases provides intercomparison between the resulting dealiased wind fields and highlights the portability of this algorithm across S-, C-, and X-band weather radar systems. Note that no additional training of the U-Net was performed prior to applying the algorithm to these other radars.

The first case is an EF5 tornado that impacted Moore, Oklahoma, on 20 May 2013, as observed by the University of Oklahoma PX-1000 research radar and the Oklahoma City TDWR (Kurdzo et al. 2015). The selected PX-1000 time is 2001:44 UTC, and the TDWR time is 2002:00 UTC. At these times, the differential velocity within the tornado via PX-1000 was approximately 65 m s$^{-1}$ (Kurdzo et al. 2015) and the tornado was producing EF4 damage in western Newcastle (Burgess et al. 2014; Atkins et al. 2014). When compared with the manual dealiasing in Figs. 2a and 2b in Kurdzo et al. (2015), the U-Net VDA accurately dealiases the broadscale mesocyclone (Fig. 8). The VDA on TDWR data also performs qualitatively well in the mesocyclone near the rear-flank gust front where there are
strong inbound radial velocities. The VDA on both radars also accurately dealiases the inflow to the supercell (areas of outbounds/red colors northwest of each radar).

Within the tornadic core circulation, the VDA has difficulty dealiasing velocities properly. There are several reasons for these difficulties. First, it should be noted that dealiasing in high-shear environments such as tornadoes is exceptionally difficult, as recently described by Feldmann et al. (2020). This is particularly true at high frequencies due to the lower Nyquist velocity. Additionally, when the Nyquist velocity is low, the wide spectrum width in tornadoes makes it difficult for the spectral estimator to determine an accurate velocity (Yu et al. 2007). In Fig. 8, a qualitative assessment shows that the VDA applied to PX-1000 is not able to dealias velocities within the core circulation, specifically in the region of outbounds (red colors). This is likely due to the highest area of spectrum width being in that region but may also be due to the existence of three folds, which this iteration of the VDA is not capable of unfolding. The VDA applied to TDWR performs almost identically, with the area in the rear flank (inbounds/blue colors) dealiased properly, but the core circulation dealiased improperly. In the case of the TDWR, the core circulation is improperly dealiased as inbounds rather than outbounds.

It is important to note that, for the TDWR, the velocity estimates were properly dealiased in the raw data because of the dual-pulse repetition time (PRT) VDA on the operational TDWR radar signal processor (Cho 2010). This dual-PRT approach had an aliasing velocity over 40 m s\(^{-1}\). To produce the images in Fig. 8, an artificial Nyquist velocity of 16.5 m s\(^{-1}\) was applied to the raw velocity data to simulate aliased velocities.

The second case is a long-lived supercell that moved across northern Illinois and Indiana on the evening of 13 June 2022, producing two brief EF0 tornadoes in the Chicago, Illinois, suburbs (https://www.weather.gov/lot/2022jun13). The posttornadic storm is shown in Fig. 9 as it passed through an area located along approximately the same radial direction from two nearby radars, a TDWR (TORD in Romeoville, Illinois; Fig. 9, top row) and a WSR-88D (KLOT in Romeoville; Fig. 9, bottom row), producing comparable radial velocity perspectives. Across both systems, the U-Net VDA performs qualitatively well, with two minor regions of errors when dealiasing the TDWR at ranges close to the radar (Fig. 9, top row). The WSR-88D is dealiased without any apparent errors (Fig. 9, bottom row).
6. Discussion, future directions, and conclusions

The results in section 5 demonstrate that DL has the potential to be an effective algorithmic tool for weather radar product development. A major challenge for supervised learning approaches like the one taken here is that they require large quantities of “truth” in order to train. In many cases (particularly with weather radar), large quantities of truth are not available, so if DL techniques are to be widely adopted in this space, alternative sources and approaches need to be leveraged.

In the case of dealiasing, this issue was overcome by utilizing an operational algorithm as truth, namely, the Level-III velocity product dealiased by the ORPG VDA. As a result, the DNN trained here can be viewed as a kind of ML “surrogate” to an already operational algorithm that is capable of generating very similar output (CSIs > 95% and RMSEs near 1.5 m s\(^{-1}\)) in a fraction of the run time, and without access to environmental wind information. These faster implementations enabled by the DL approach could be leveraged in real-time settings where traditional dealiasing techniques fail to keep up with certain rapid scanning strategies. For example, if real-time dealiasing were desired for an extremely rapidly updating research radar [e.g., as fast as 2-s updates such as presented in Pazmany et al. (2013)], this implementation of a VDA would be capable of keeping up in real time with relatively modest hardware. The DL models may also be useful in research settings by allowing massive velocity datasets to be dealiased in minutes, rather than hours or days using traditional approaches.

Last, more complex VDAs such as higher-dimensional techniques could be trained using DL in the future in order to utilize their approaches in real time.

Another advantage of the ORPG surrogates is portability. Modern DL frameworks and deployment frameworks simplify the importing and application of trained models to new datasets without the need for additional software installations and dependencies. In addition, the ORPG code is written exclusively for the WSR-88D network, with little ability to port algorithms to other types of radar (i.e., different vendors, frequencies, beamwidths, etc.). This flexibility provided by the U-Net allows for simpler application of an ORPG-like VDA to new radar types without the need for the significant data reformatting required by the traditional ORPG algorithm. Improved performance on other radar types was also gained
by training with Nyquist velocities that were lower than that commonly set by WSR-88D operators. This enabled impressive performance on other unseen radars (TDWR; PX-1000) despite the fact that the model was not trained directly on these systems. The prospect of porting ORPG-like algorithms to other radar systems may provide benefits to weather radar networks outside of the United States where these algorithms are not currently available. Further performance improvements on non-WSR-88D data can potentially be enabled using transfer learning by fine-tuning the network developed here.

Note that the VDA is far from the only ORPG product that is computationally complex. As of Build 20, the ORPG maintains a suite of over 150 products, which must all be run in real time and keep up with the fastest scan rates. This plethora of algorithms may need to be run in a much more expedient manner in the future with faster scan rates, such as those discussed in Kollias et al. (2022) and Palmer et al. (2022). Additionally, other radar systems may not have access to the hardware available for the ORPG. For example, research radars may want to implement WSR-88D algorithms in real time but do not have the computational capabilities.

Given the results of this work, a natural question to ask is how this approach can be leveraged for other weather radar product enhancements. A whole suite of Level-III radar products has been successfully developed using traditional signal, image, and video processing techniques, and these methods have their constraints and weaknesses that can lead to difficulty in both implementation and operations. Additionally, like any algorithm, many of the ORPG algorithms suffer from false alarms, missed detections, and/or relatively noisy interpretations of untruthed data [e.g., the hydrometeor classification algorithm (HCA) (Park et al. 2009)]. However, ORPG algorithms, in many cases, are considered state of the art, and are desirable tools to have available to other radar systems. One way to explore this possibility is to implement other ML-based surrogates to ORPG algorithms. ML-based versions of the ORPG algorithms could result in several advantages, ranging from faster compute times to (potentially) improved accuracy through continued fine-tuning of ML surrogates using additional datasets.

Overall, in this work, a DL approach was developed for two-dimensional velocity dealiasing of weather radar data. To construct a training dataset, archives of Level-II and Level-III WSR-88D radial velocity products were combined to create a large dataset of over 100,000 dealiased velocity PPIs. To create aliased/dealiased pair samples for training, these scans were artificially folded using a known Nyquist velocity.

FIG. 9. Application of the U-Net algorithm to two different test datasets: (top) artificially folded TDWR data at 0003 UTC 14 Jun 2022 and (bottom) NEXRAD Level-II velocity on the same date at 0001 UTC. The solid line denotes a smoothed 30-dBZ reflectivity factor contour, and the dashed lines indicate the primary areas of velocity aliasing that were recovered. The U-Net VDA adequately corrects the aliased velocities in both cases, with a small area of errors near or just north of the TDWR. The axis labels are equivalent to the distance from each respective radar in kilometers.
A U-Net model was trained using this dataset to output the correct fold number at each point in an input velocity image. The U-Net model developed here utilized several convolutional layers of different sizes to ensure it properly learned both global and local structure in the input image. Once trained, the U-Net model generated similar results to the operational ORPG algorithm from which it was trained based on a test set of held-out WSR-88D data. Encouragingly, the model also performed reasonably well on radars that were not included in the training dataset, including the TDWR and PX-1000 radars.

Future work will continue to explore potential DL applications in radar meteorology, including the development of other ORPG surrogate algorithms. These techniques have the potential to improve existing radar products and to enable ORPG product generation on non-WSR-88D radars.

Acknowledgments. The authors thank Mike Donovan, Betty Bennett, and Mike Istok for helpful discussions about the WSR-88D ORPG. The authors also thank the anonymous reviewers for their valuable insights and suggestions. This material is based upon work supported by the Department of the Air Force under Air Force Contract FA8702-15-D-0001. Any 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.

Data availability statement. Trained model weights, sample data, and code for running the U-Net VDA are available at GitHub (https://github.com/mit-lab/unet-vda). Source codes for ORPG are available online (https://www.weather.gov/code88d/). WSR-88D data used in this work are publicly available from AWS Open Data Registry (https://registry.opendata.aws/nexrad/) and Google Cloud Storage (https://cloud.google.com/storage/docs/public-datasets/nexrad). TDWR data are publicly available from radar_px1000.html (https://arrc.ou.edu/storage/docs/public-datasets/nexrad). TDWR data are publicly available from ORPG (https://www.weather.gov/code88d/). The PX-1000 (https://arrc.ou.edu/radar_px1000.html) is maintained and operated by the Advanced Radar Research Center of the University of Oklahoma, and data can be requested by contacting data@arrc.ou.edu.

REFERENCES


