Can We Integrate Spatial Verification Methods into Neural Network Loss Functions for Atmospheric Science?

RYAN LAGERQUIST[a,b] AND IMME EBERT-UPHOFF[a,c]

[a] *Cooperative Institute for Research in the Atmosphere, Colorado State University, Fort Collins, Colorado*
[b] *NOAA/ESRL Global Systems Laboratory, Boulder, Colorado*
[c] *Department of Electrical and Computer Engineering, Colorado State University, Fort Collins, Colorado*

(Manuscript received 15 March 2022, in final form 12 September 2022)

ABSTRACT: In the last decade, much work in atmospheric science has focused on spatial verification (SV) methods for gridded prediction, which overcome serious disadvantages of pixelwise verification. However, neural networks (NN) in atmospheric science are almost always trained to optimize pixelwise loss functions, even when ultimately assessed with SV methods. This establishes a disconnect between model verification during versus after training. To address this issue, we develop spatially enhanced loss functions (SELF) and demonstrate their use for a real-world problem: predicting the occurrence of thunderstorms (henceforth, "convection") with NNs. In each SELF we use either a neighborhood filter, which highlights convection at scales larger than a threshold, or a spectral filter (employing Fourier or wavelet decomposition), which is more flexible and highlights convection at scales between two thresholds. We use these filters to spatially enhance common verification scores, such as the Brier score. We train each NN with a different SELF and compare their performance at many scales of convection, from discrete storm cells to tropical cyclones. Among our many findings are that (i) for a low or high risk threshold, the ideal SELF focuses on small or large scales, respectively; (ii) models trained with a pixelwise loss function perform surprisingly well; and (iii) nevertheless, models trained with a spectral filter produce much better-calibrated probabilities than a pixelwise model. We provide a general guide to using SELFs, including technical challenges and the final Python code, as well as demonstrating their use for the convection problem. To our knowledge this is the most in-depth guide to SELFs in the geosciences.

SIGNIFICANCE STATEMENT: Gridded predictions, in which a quantity is predicted at every pixel in space, should be verified with spatially aware methods rather than pixel by pixel. Neural networks (NN), which are often used for gridded prediction, are trained to minimize an error value called the loss function. NN loss functions in atmospheric science are almost always pixelwise, which causes the predictions to miss rare events and contain unrealistic spatial patterns. We use spatial filters to enhance NN loss functions, and we test our novel spatially enhanced loss functions (SELF) on thunderstorm prediction. We find that different SELFs work better for different scales (i.e., different-sized thunderstorm complexes) and that spectral filters, one of the two filter types, produce unexpectedly well calibrated thunderstorm probabilities.

KEYWORDS: Deep convection; Filtering techniques; Fourier analysis; Neural networks; Optimization; Machine learning

1. Introduction

Many problems in atmospheric science involve predicting on a spatial grid. Traditionally, gridded predictions have been verified pixelwise, where the prediction at pixel P is compared only with the observation at P, and vice versa. Pixelwise verification has three main disadvantages. The first is the double-penalty problem (Gilleland et al. 2009), which is most prevalent when predicting a spatially rare event—that is, one that occurs at only a small fraction of pixels, like heavy rainfall. If the predictions have a small offset (enough that predicted and observed event areas do not line up), even if the predictions are otherwise perfect, verification scores are very poor. This is because pixels with the predicted event are counted as false positives, while pixels with the observed event are counted as false negatives. The combination of false positives and false negatives is the "double penalty," and this encourages the model not to predict spatially rare events at all. The second disadvantage is that pixelwise verification scores are not well correlated with the perceptual (i.e., subjective) quality of the prediction. For example, Fig. 2 of Wang and Bovik (2009) shows that pixelwise mean square error (MSE) is often much worse for images with better perceptual quality. Third, pixelwise verification encourages models not to make predictions with sharp gradients—even if the observations contain sharp gradients—because the sharp gradients will likely not be at the exact right location, leading to high pixelwise error. By avoiding sharp gradients, models produce predictions that are too smooth or "blurry" [e.g., the "CNN SR" in Figs. 3 and 7 of Stengel et al. (2020)].

Supplemental information related to this paper is available at the Journals Online website: https://doi.org/10.1175/AIES-D-22-0021.s1.

Corresponding author: Ryan Lagerquist, ralager@colostate.edu

DOI: 10.1175/AIES-D-22-0021.1 e220021

a. Spatial verification

To address these problems, much recent work has focused on spatial verification (SV). One of the most popular SV methods in atmospheric science is the fractions skill score (FSS; Roberts and Lean 2008), which compares gridded predictions and observations of a binary event, such as heavy rainfall, where the only possible values are 1 (yes) and 0 (no). The FSS includes a neighborhood distance, so, when focusing on pixel P, it actually compares the predicted and observed event fractions in the surrounding patch. This patch may be 3×3, 5×5, 7×7, and so on. For a large enough patch, the FSS avoids the double-penalty problem. A similar SV method is popular in the computer-vision literature, called the structural similarity (SSIM) index (Wang et al. 2004; Wang and Bovik 2009). The SSIM index can be used to verify a regression model (one that predicts continuous values, such as rainfall amount in mm) or a classification model. At each pixel P, the SSIM index is the product of three similarities computed over the surrounding patch: the similarity of raw values ("luminances" in Wang and Bovik 2009), contrasts (based on the variance in each image), and structures (based on the covariance between images). The FSS has been widely adopted in atmospheric science (Weusthoff et al. 2010; Sobash et al. 2011; Mittermaier et al. 2013; Bachmann et al. 2018; Ahmed et al. 2019; Loken et al. 2019; Qian and Wang 2021), as has the SSIM index in computer science (Johnson et al. 2016; Zhao et al. 2017; Hammernik et al. 2017; Ledig et al. 2017; Snell et al. 2017; Wang et al. 2018).

SV methods in atmospheric science go far beyond the FSS. Gilleland et al. (2009) provided an overview, splitting SV methods into four categories: neighborhood, scale-separation, feature-based, and field-deformation methods. Neighborhood (or "fuzzy") methods, like the FSS and SSIM index, use a patch centered at pixel P to compare the predictions and observations around P. Scale-separation methods apply a spectral filter (i.e., a low-pass, high-pass, or bandpass filter implemented with either Fourier or wavelet decomposition) to both the predictions and observations, thus isolating patterns over the desired range of physical scales, followed by pixelwise verification. Feature-based methods match high-level features (e.g., cold fronts) between the predictions and observations, while field-deformation (or "morphing") methods determine how much manipulation is needed to make the predictions and observations match.

b. From spatial verification to spatially enhanced loss functions

There is a growing recognition that physically meaningful loss functions are needed in the geosciences (Karpatne et al. 2017; Willard et al. 2020; Beucler et al. 2021; Lagerquist et al. 2021b). Neural networks (NN) have become a common tool in atmospheric science and the geosciences at large. For gridded prediction, a popular NN architecture is the U-net (Chen et al. 2020; Kumler-Bonfanti et al. 2020; Sadeghi et al. 2020; Sha et al. 2020a,b; Han et al. 2021; Lagerquist et al. 2021a,b), invented by Ronneberger et al. (2015). Lagerquist et al. (2021a) contains an explanation of U-nets for atmospheric scientists. NNs, including U-nets, are trained to minimize a loss function, which measures the error between the predictions and observations. Despite much recent work in SV, NNs in the geosciences are almost always trained with pixelwise loss functions. This establishes a disconnect between the model verification during and after training—that is, the model is trained to optimize pixelwise performance but is verified on spatial performance.

To our knowledge, spatially enhanced loss functions (SELF) appear in only a few geoscience applications. First, Zhang et al. (2008) used Baddeley's Δ metric, a field-deformation method, for a climate application: to predict the region of the Americas exceeding a given temperature change from the 1980s to 1990s. Second, Gilleland (2021) developed new field-deformation metrics and suggested their use as loss functions in models that predict a spatial-exceedance region, like that of Zhang et al. (2008). One advantage of these new metrics is that they handle the case where all values in the predicted or observed field are the same—for example, 0 for no convection. Third, Heim and Avery (2019) used the image Euclidean distance (IMED; Wang et al. 2005) to detect switches in the Kuroshio, a major ocean current east of Japan, between an elongated and contracted state. They framed the problem as anomaly detection in time series and used a convolutional neural network to solve the problem. The IMED is similar to the pixelwise MSE but accounts for spatial autocorrelation between nearby pixels. Fourth, Stengel et al. (2020) used a composite loss function to upsample fields of wind velocity and solar irradiance. Their composite loss function is pixelwise MSE plus "adversarial loss," which is provided by the discriminator of a generative adversarial network (GAN; Goodfellow et al. 2014; section 20.10.4 of Goodfellow et al. 2016). The discriminator learns high-level features of images, which are more closely related to human perception of image similarity than are pixelwise measures. Stengel et al. (2020) thus called the adversarial loss a "perceptual loss," and their Figs. 3 and 7 show that it leads to more realistic upsampled images than does the pixelwise MSE.

Besides perceptual loss functions, some work in computer science and remote sensing uses local spatial gradients to spatially enhance the loss function (e.g., Eigen and Fergus 2015). This is most often done by using the SSIM index as the loss function (Zhao et al. 2017; Hammernik et al. 2017; Snell et al. 2017; Harder et al. 2020).

c. Emphasis and scope of this study

In Ebert-Uphoff et al. (2021), our group provided an introduction to custom loss functions in the geosciences, including the idea of using the FSS as a loss function. This idea was tested for the first time in Lagerquist et al. (2021a, henceforth L21a), for the problem of forecasting convection at 0–2-h lead times. Specifically, L21a used the 9-by-9-pixel FSS as a loss function and demonstrated that U-nets trained with the 9-by-9 FSS outperform those trained with a pixelwise loss function. However, L21a explored only this one SELF, whereas in this paper we focus entirely on exploring many SELFs in depth. Specifically, from the four types of SV methods discussed in Gilleland et al. (2009), we incorporate two in

loss functions: neighborhood and scale-separation methods. We turn verification scores common in atmospheric science, such as the Brier score and FSS, into both neighborhood and scale-separation loss functions. The latter use spectral filtering, via Fourier or wavelet decomposition, to separate scales.[1] Our prediction task is the same as in L21a, but we focus only on the 1-h lead time.

We address four scientific questions in this study: 1) What are the options for neighborhood and scale-separation loss functions? 2) How can these loss functions be implemented? 3) How do these loss functions influence the results? 4) For a given geoscience application, can we use expert knowledge to determine which loss functions should (not) be used? To our knowledge, this paper is the first in-depth exploration of SELFs in the geosciences. We have included code for all SELFs (see data availability statement below), using the Keras and TensorFlow libraries in Python (Chollet et al. 2015).

2. Sample application

This section briefly introduces our sample application: the prediction task (1-h forecasting of convection), input data, and an explanation of the U-net NN architecture. See L21a for further details.

As predictors for convection, we use a time series of brightness-temperature maps from the *Himawari-8* satellite, with seven spectral bands (Figs. 1a–g) and three lag times. The seven spectral bands—6.25, 6.95, 7.35, 8.60, 10.45, 11.20, and 13.30 μm—are all in the infrared portion of the electromagnetic spectrum, so our U-nets can be used during both day and night. The three lag times are 0, 20, and 40 min before the forecast-issue time t_0. As targets (treated as correct answers during U-net-training), we use a convection mask at t_0 + 1 h. The mask is produced by applying a modified version of Storm-Labeling in 3 Dimensions (SL3D; Starzec et al. 2017), an echo-classification algorithm, to radar data from Taiwan (Fig. 1h). The mask contains 1 at pixels with a thunderstorm and 0 elsewhere (Fig. 1i). Both the satellite and radar data are provided by the Taiwan Central Weather Bureau on a 0.0125° grid. We verify the U-nets only at pixels within 100 km of the nearest radar (Fig. 1i), deemed to have adequate coverage for detecting convection.

Our chosen NN architecture is the U-net, which excels at gridded prediction (Ronneberger et al. 2015). A U-net contains four components (Fig. 2): convolutional layers, pooling (downsampling) layers, upsampling layers, and skip connections. The convolutional layers detect spatial and multivariate features (here, spatial features involving many spectral bands and lag times); all other components allow different convolutional layers to detect features at different scales. Inputs to the first layer are raw predictors (here, brightness temperatures), and inputs to all other layers are transformed versions of the raw predictors, called feature maps. Each convolutional layer is followed by a nonlinear activation function; without these, the U-net would learn only linear relationships. Our chosen activation function is the leaky rectified linear unit (ReLU; Maas et al. 2013). Each pooling layer downsamples the feature maps to a lower spatial resolution, here using a 2-by-2 maximum filter. Hence, on the left (downsampling) side of Fig. 2, grid spacing varies from 0.0125° at the top to 0.4° at the bottom. As the spatial resolution decreases, the number of feature maps ("channels") increases to offset the loss of spatial information. Each upsampling layer upsamples the feature maps to a higher spatial resolution, using interpolation followed by convolution. Hence, on the right (upsampling) side of Fig. 2, grid spacing varies from 0.4° at the bottom to 0.0125° at the top. As the spatial resolution increases, the number of channels decreases, terminating here with one channel (convection probability). Skip connections carry high-resolution information from the downsampling side of the U-net directly to the upsampling side, which is crucial because upsampling alone is not sufficient to recover the spatial information lost during downsampling. Skip connections include convolutional layers, as explained in L21a. In all convolutional layers (on the downsampling side, on the upsampling side, and in the skip connections), weights in the convolutional kernels are learned during training. The final set of weights is that which minimizes the loss function. For more details on the U-net architecture, including the inner workings of all components, see section 3b of L21a.

As mentioned above, we verify the U-nets only at pixels < 100 km from the nearest radar. Thus, at pixels > 100 km from the nearest radar, there is no target value (i.e., neither 0 nor 1 in the convection mask). To handle missing target values, we train each U-net with 205-by-205 radar-centered patches of the full 881-by-921 grid. At inference time, to apply a trained U-net to the full grid, we slide the 205-by-205 window around the full grid. For more details on these training and inference methods, see sections 4a and 4b of L21a.

3. Neighborhood loss functions

This section defines the neighborhood loss functions used and discusses how we implement them in code.

a. *Definition*

A neighborhood loss function involves a mean or maximum filter, taken over a square neighborhood of pixels. The neighborhood dimensions are always odd (e.g., 3×3 or 5×5), so that the center aligns with one pixel. A key motivation for neighborhood loss functions is that they avoid the double-penalty problem (discussed in section 1), because they always expand convective regions (Fig. 3), thus increasing overlap between the predictions and observations. Relative to scale-separation methods, the main advantages of neighborhood methods are ease of implementation (cf. sections 3b and 4c) and transparency. To the second point, it is easy to guess how a neighborhood filter will change the original data (Fig. 3), but it is not so for the filters involved in scale separation (Fig. 4).

[1] However, for reasons that will become clear in section 4c, it is better to perform spectral filtering outside the loss function.

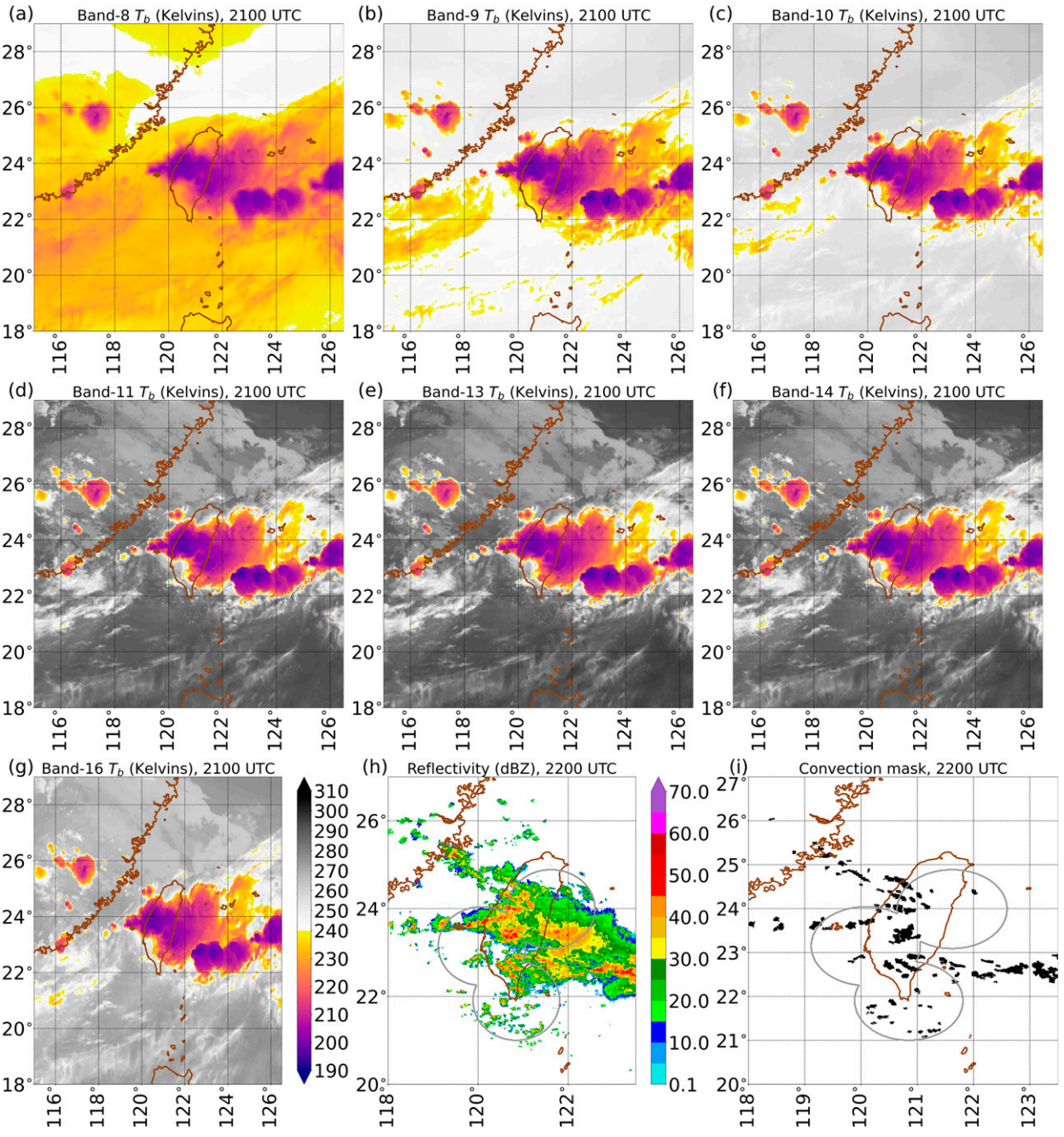

FIG. 1. Example of input data for U-net, featuring predictors valid at 2100 UTC 2 Jun 2017 and targets valid 1 h later, at 2200 UTC. (a)–(g) Brightness temperature [K; color bar in (g)] in each spectral band, used as predictors. (h) Composite (column maximum) radar reflectivity. (i) Convection mask. The black dots are pixels with convection, according to the SL3D algorithm used to create targets. Gray-outlined circles in (h) and (i) show the 100-km range ring around all but the northernmost radar. Only pixels inside these range rings are used to train and verify the U-nets. The northernmost radar is omitted from training and verification because of the data-quality issues discussed in L21a.

Although neighborhood loss functions are easy to code and appropriate for many atmospheric-science problems, only a few studies have done this, and they use only the FSS (L21a; Earnest et al. 2022; Justin et al. 2022). We go beyond the FSS and turn five other common verification scores into neighborhood loss functions: the Brier score, intersection over union (IOU), Dice coefficient, critical success index (CSI), and cross entropy. Table 1 defines the traditional and neighborhood-based version of all six scores. The traditional version is pixelwise for all scores except the FSS, which is already neighborhood

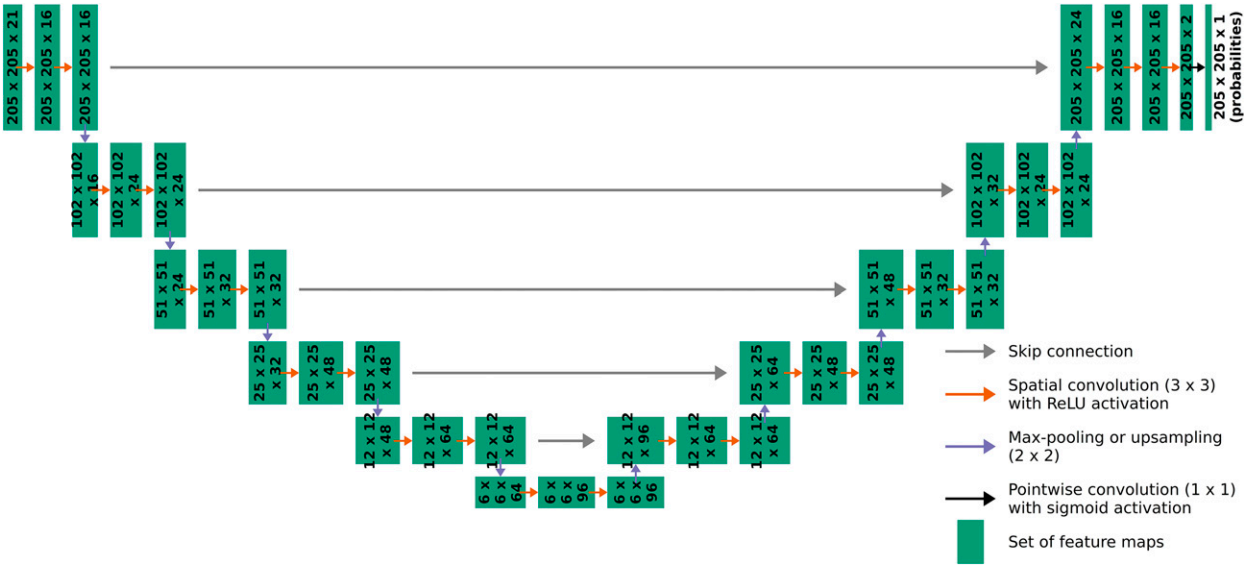

FIG. 2. U-net architecture. The left side is the downsampling side, and the right side is the upsampling side. In each set of feature maps, the numbers are $N_{\text{rows}} \times N_{\text{columns}} \times N_{\text{channels}}$. The 21 input channels at the top left are the predictor variables, i.e., brightness temperatures from 7 spectral bands at 3 lag times. Spatial-convolution filters have dimensions of 3 rows × 3 columns; pixelwise-convolution filters have dimensions of 1 × 1; pooling and upsampling windows have dimensions of 2 × 2.

based.[2] Variables in Table 1 are defined below, along with an intuitive definition of each score. For positively oriented scores s (where higher values are better; all except the Brier score and cross entropy), we use $1 - s$ as the loss function, since loss functions are negatively oriented. We do the same for scale-separation loss functions (section 4).

In the traditional Brier score, G is the number of pixels, p_g is the predicted event probability at pixel g, and y_g is the observation (0 or 1) at g. The traditional Brier score is the MSE adapted for binary classification. In the neighborhood Brier score, r is the neighborhood half-width and $y_g^{\max}(r)$ is the maximum observation in the neighborhood (Fig. 5). Thus, $y_g^{\max}(r)$ is 1 if the event occurs anywhere in the neighborhood and 0 otherwise.

In the FSS, $\overline{p_g}(r)$ is the mean probability, and $\overline{y_g}(r)$ is the mean observation, in a neighborhood of half-width r centered at pixel g. The numerator is the actual sum of squared errors (SSE) between mean-filtered predictions and observations, while the denominator is the reference SSE—the maximum SSE possible given the two fields, which occurs if they are completely misaligned.

Note that the FSS traditionally involves binary predictions (0s and 1s) but a U-net, like most machine learning models for classification, outputs probabilities ranging continuously over [0, 1]. These probabilities *could* be transformed to binary predictions by discretization, that is, thresholding the probabilities. However, the best threshold is not clear and is typically not the intuitive guess of 0.5. One *could* find the best probability threshold (i.e., that which minimizes the loss function) every time the loss function is evaluated, but this operation would be computationally expensive and would involve the nondifferentiable minimum operator.[3] Thus, for all scores other than the Brier score and cross entropy (which are already probabilistic), we replace binary predictions in the traditional definition with probabilities. We do the same for scale-separation loss functions (section 4). This changes the physical meaning of the FSS, which was originally meant to compare the forecast and observed event coverage (here, percentage of convective pixels in the neighborhood). Here, we compare the observed event coverage with the spatial sum of forecast event probabilities, rather than the forecast event coverage. In addition to the technical challenges that come with thresholding, we speculate that using raw probabilities in the loss functions allows the NNs to achieve better performance. The magnitudes of the NN-forecast probabilities contain information, and thresholding destroys much of this information.

In the traditional IOU, $\max(p_g, y_g)$ is the maximum between the prediction and observation at pixel g. The numerator is the intersection between the two fields (where the event both occurs and is predicted), and the denominator is the union (where the event either occurs or is predicted). In the neighborhood IOU, $y_g^{\max}(r)$ is defined as in the neighborhood Brier score. Note that we use only the positive-class IOU: for class 1 (convection), rather than class 0 (no convection).

The Dice coefficient is similar to the IOU; the main difference is that the denominator is the area of the full domain,

[2] The FSS can be turned into a pixelwise score by using a single pixel, i.e., a 1-by-1 neighborhood. In this case, the equation in Table 1 becomes $1 - [\sum_{g=1}^{G}(p_g - y_g)^2 / \sum_{g=1}^{G}(p_g^2 + y_g^2)^2]$.

[3] All loss functions must be differentiable, as discussed further in section 4c.

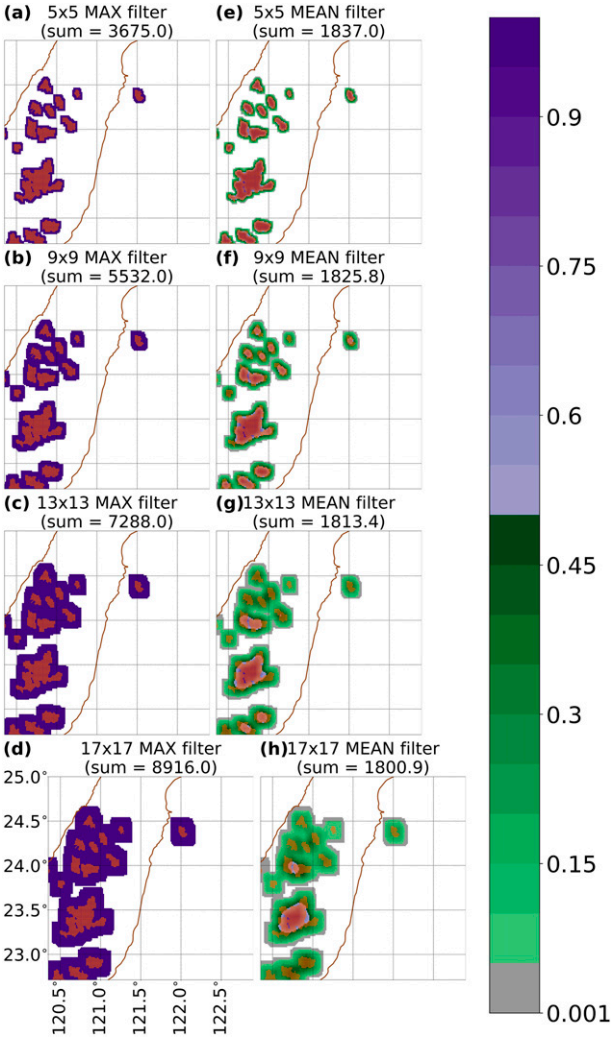

FIG. 3. Effect of neighborhood filtering on target fields. Orange dots on top of the color maps show the target field before filtering (convective pixels are marked with an orange dot, whereas nonconvective pixels have no orange dot); the color maps show the target field after filtering. Before filtering, all target values are exactly 0 or 1; after filtering, target values range continuously from 0 to 1. The sum in each panel title is the sum over all pixels after filtering. This figure shows 4 of the 8 neighborhood sizes used in the experiment (see Table 3 for the full list). (a)–(d) Target fields, valid at 2200 UTC 2 Jun 2017, after maximum filter, used for all scores except FSS (Table 1). (e)–(h) As in (a)–(d), but with mean filter, used for FSS.

rather than the intersection between predicted and observed events. For spatially rare events, if defined only for the positive class, the Dice coefficient is typically very small, because the full domain is much larger than either the predicted or observed event areas. Thus, we use the all-class Dice coefficient. In the traditional Dice coefficient, $1 - p_g$ is the negative-class probability and $1 - y_g$ is the negative-class label (0 if the event occurs; 1 if it does not). The definition of $1 - y_g^{\max}(r)$ is analogous but with the convection labels maximum-filtered over a neighborhood with half-width r.

The CSI is based on the contingency table for binary classification, which traditionally has four elements: the number of true positives a, false positives b, false negatives c, and true negatives d. However, in the neighborhood setting, there are two types of true positives: prediction and observation oriented (Fig. 5). The National Weather Service verifies tornado warnings in a similar setting (Brooks 2004), where the contingency table has the following elements: number of prediction-oriented true positives a_{pred}, observation-oriented true positives a_{obs}, false positives b, and false negatives c. Although probability of detection (POD) and SR are defined differently in the traditional and neighborhood settings, $\text{CSI}^{-1}(r) = \text{POD}^{-1}(r) + \text{SR}^{-1}(r) - 1$ holds in both settings (Brooks 2004; Roebber 2009). Because we do not discretize probabilities from the U-nets, we use a probabilistic form of the contingency table (Fig. 5).

Cross entropy (often called "logarithmic loss" or "log loss") is a commonly used loss function in machine learning and is not limited to atmospheric-science applications. For a perfect model, $p_g = 1$ wherever $y_g = 1$, causing the numerator to be $1\log_2(1) + 0\log_2(0) = 1(0) + 0(-\infty) = 0$. Also, for a perfect model, $p_g = 0$ wherever $y_g = 0$, causing the numerator to be $0\log_2(0) + 1\log_2(1) = 0(-\infty) + 1(0) = 0$. Hence, the perfect cross entropy is zero.

b. *Implementation*

Neighborhood loss functions are easy to implement in NN libraries, because the mean and maximum filters used can be achieved via convolution and pooling, respectively, which are predefined NN operations and are differentiable. Hence, we put neighborhood filters directly in the loss function, as illustrated in Fig. 6. (See the data availability statement below for a link to the code.)

4. Scale-separation loss functions

Relative to neighborhood methods, the main advantage of scale-separation methods is their ability to focus on a specified *range* of scales, with an upper and lower bound. Neighborhood filters always preserve information at the larger scales and degrade it at the smaller scales, but the spectral filters used in scale separation preserve information at scales inside the desired range and degrade it at scales outside the desired range. We explore both Fourier and wavelet decomposition for scale separation, because (i) they are the most common methods and (ii) they have very different properties, as discussed later in this section, and it is not obvious how these properties will influence what an NN learns.

The remainder of this section begins with a discussion of previous work combining Fourier and wavelet decomposition with NNs, then defines the scale-separation loss functions that we use, then discusses how we implement them in code.

a. *Previous work combining Fourier and wavelet decomposition with neural networks*

To our knowledge we are the first to explore using Fourier or wavelet decomposition in loss functions, but in general, the

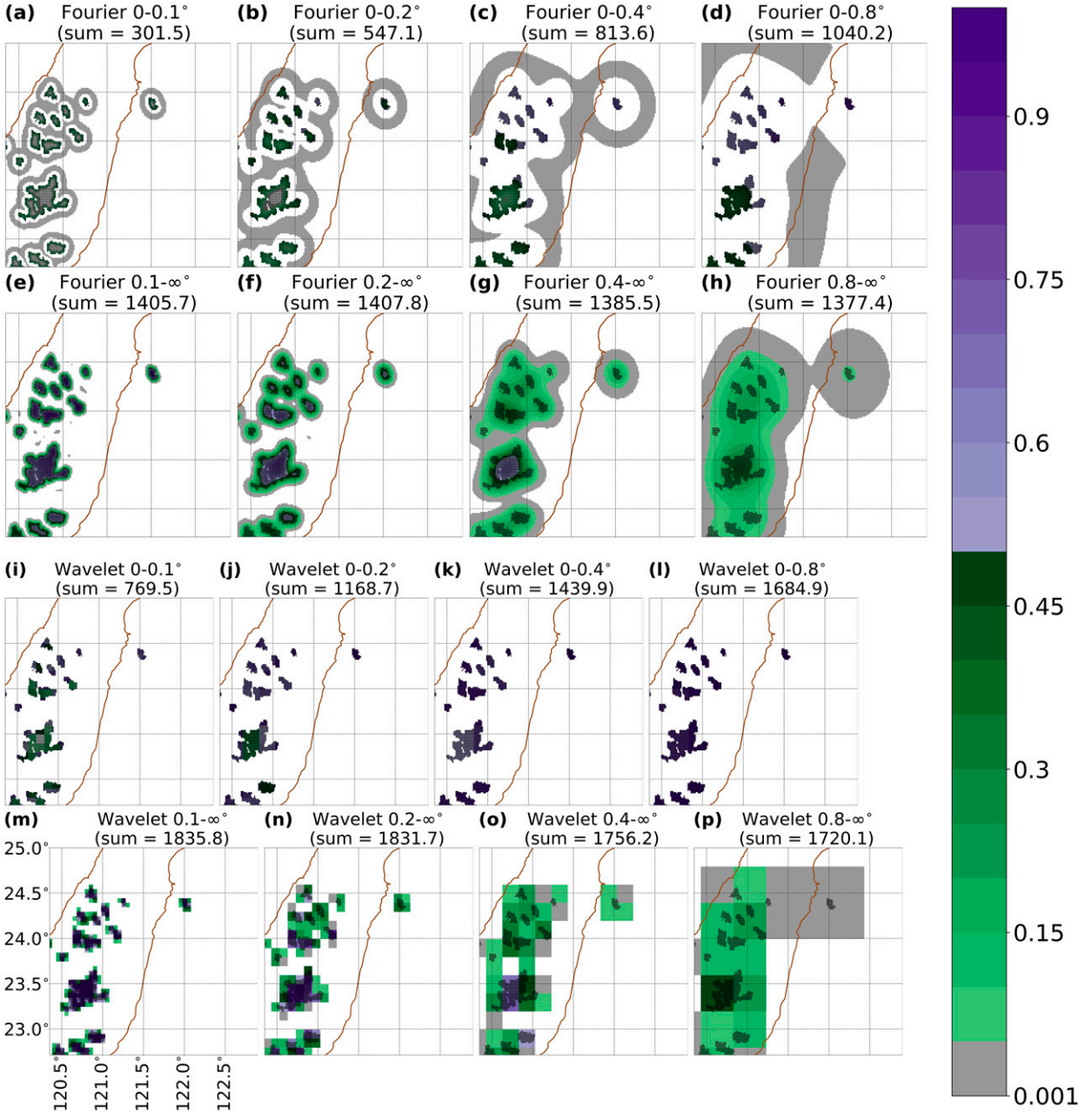

FIG. 4. Effect of spectral filtering on target fields. Formatting is as in Fig. 3, except that convective pixels are marked with a black dot rather than an orange dot. (We use black dots henceforth in the paper; we used orange dots in Fig. 3 because black did not contrast enough with the large areas of dark purple.) This figure shows 8 of the 16 wavelength bands used in the experiment (see Table 3 for the full list). (a)–(h) Target fields, valid at 2200 UTC 2 Jun 2017, after a spectral filter is implemented via Fourier decomposition. (i)–(p) As in (a)–(h), but with wavelet decomposition.

idea of using Fourier and wavelet decomposition in NNs is not new.

Many studies have used Fourier decomposition to accelerate the training of CNNs, leveraging the fact that convolution in the spatial domain is equivalent to elementwise multiplication in the Fourier domain (Mathieu et al. 2014; Rippel et al. 2015; Pratt et al. 2017). Also, Lee-Thorp et al. (2021) replaced the self-attention sublayer (which contains many learned parameters) in natural-language-processing models with a Fourier transform (which contains no learned parameters), achieving comparable accuracy in much less computing time. Fujieda et al. (2017) replaced pooling layers in CNNs (which keep only the low-frequency part of an image) with wavelet-decomposition layers (which keep both the high- and

TABLE 1. Verification scores used in both neighborhood and scale-separation loss functions. Each score is defined for one time step; to obtain the score for multiple time steps, compute the score for each time step individually and then average. For a scale-separation loss function, the traditional (pixelwise) definition of the score is applied to the spectrally filtered predictions and observations. The range of possible values is $[0, \infty)$ for cross entropy and $[0, 1]$ for all other scores. Variables in the equations, as well as an intuitive definition of each score, are provided in the main text.

Score	Traditional definition	Neighborhood-based definition	Optimal value
Brier score	$\frac{1}{G}\sum_{g=1}^{G}(p_g - y_g)^2$	$\frac{1}{G}\sum_{g=1}^{G}[p_g - y_g^{\max}(r)]^2$	0
FSS	Not applicable	$1 - \frac{\sum_{g=1}^{G}[\overline{p_g}(r) - \overline{y_g}(r)]^2}{\sum_{g=1}^{G}[\overline{p_g}^2(r) + \overline{y_g}^2(r)]}$	1
IOU	$\frac{\sum_{g=1}^{G} p_g y_g}{\sum_{g=1}^{G}\max(p_g, y_g)}$	$\frac{\sum_{g=1}^{G} p_g y_g^{\max}(r)}{\sum_{g=1}^{G}\max[p_g, y_g^{\max}(r)]}$	1
Dice coef	$\frac{\sum_{g=1}^{G} p_g y_g + \sum_{g=1}^{G}(1-p_g)(1-y_g)}{G}$	$\frac{\sum_{g=1}^{G} p_g y_g^{\max}(r) + \sum_{g=1}^{G}(1-p_g)[1-y_g^{\max}(r)]}{G}$	1
CSI	$\frac{a}{a+b+c}$	$\mathrm{CSI}^{-1}(r) = \mathrm{POD}^{-1}(r) + \mathrm{SR}^{-1}(r) - 1$, where $\mathrm{POD}(r) = \frac{a_{\mathrm{obs}}(r)}{a_{\mathrm{obs}}(r) + c(r)}$ and $\mathrm{SR}(r) = \frac{a_{\mathrm{pred}}(r)}{a_{\mathrm{pred}}(r) + b(r)}$	1
Cross entropy	$\frac{\sum_{g=1}^{G}[y_g \log_2(p_g) + (1-y_g)\log_2(1-p_g)]}{G}$	$\frac{\sum_{g=1}^{G}\{y_g^{\max}(r)\log_2(p_g) + [1-y_g^{\max}(r)]\log_2(1-p_g)\}}{G}$	0

low-frequency parts), allowing for better texture classification. Li et al. (2020a) replaced spatial pooling in CNNs with spectral truncation (filtering of wavelet coefficients), which is better at separating the signal and noise (i.e., desired and undesired scales). Last, some works have used decomposition methods to transform predictors from spatial images to spectral (Fourier or wavelet) coefficients, then trained NNs with only the coefficients, achieving similar accuracy to training with spatial images but in much less computing time (e.g., Kiruluta 2017). Other examples of training NNs with spectral coefficients include at least two geoscience applications: predicting daily precipitation (Partal et al. 2015) and daily river discharge (Gürsoy and Engin 2019).

Of significant interest to atmospheric science is the recent development of the neural operator and Fourier neural operator (FNO), which are special types of NN architectures. Neural operators (Lu et al. 2019; Li et al. 2020b) can learn mappings between infinite-dimensional function spaces (e.g., continuous fields), whereas traditional NNs can learn mappings only between finite-dimensional Euclidean spaces (e.g., images with values at discrete grid points). This means that neural operators can learn an entire family of partial differential equations (PDE) in a grid-agnostic manner. However, neural operators are much slower than numerical PDE-solvers. Li et al. (2021) accelerated neural operators with Fourier transforms, resulting in FNOs. Li et al. (2021) demonstrated the impressive capabilities of FNOs by modeling the Navier–Stokes equation in turbulent flow. FNOs have since shown impressive skill in solving PDEs for physical problems, including atmospheric science. FNOs are a key component of FourCastNet (Pathak et al. 2022), arguably the most accurate machine learning–based numerical weather prediction (NWP) model to date. Furthermore, FNOs are several orders of magnitude faster than traditional PDE-solvers, allowing larger NWP ensembles (e.g., thousands of members) to be run with existing computational resources. Thus, FNOs will likely play a major role in future NWP-model development.

While FNOs use Fourier transforms solely to increase the speed of NN operations, our purpose for using Fourier transforms (and wavelet transforms) is to implement image-filtering in spectral space—not to increase speed.

b. Definition

The traditional verification scores from Table 1, which we turn into neighborhood loss functions (section 3), we also turn into scale-separation loss functions. For all scores except the FSS, the equation used for the scale-separation loss function is the traditional equation in Table 1. For the FSS, whose traditional definition is already neighborhood based, we use a neighborhood of 1×1 pixels. Thus, the equation for every scale-separation loss

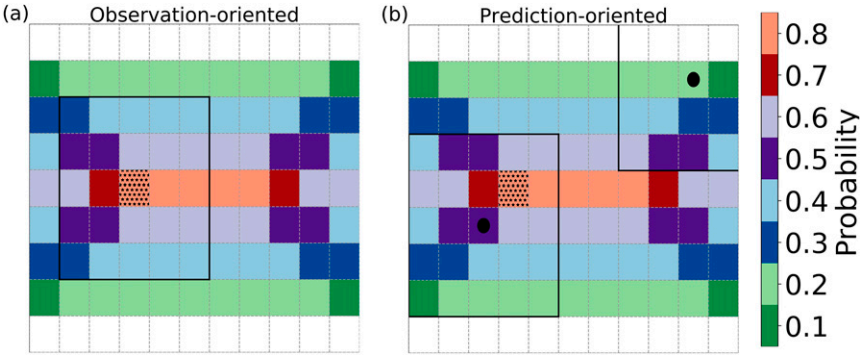

FIG. 5. Model verification with a 5-by-5 neighborhood (i.e., half-width of 2 pixels). The color map shows predicted probabilities; convection is observed at the one stippled grid cell and nowhere else. (a) Observation-oriented evaluation, where the central pixel is the one with observed convection. The maximum probability in the neighborhood is 0.8, so this situation contributes 0.8 to a_{obs} (observation-oriented true positives) and $1 - 0.8 = 0.2$ to c (false negatives). (b) Prediction-oriented verification, shown for two central pixels (marked with circles). For the bottom-left pixel (probability of 0.5), convection is observed in the neighborhood, so this situation contributes 0.5 to a_{pred} (prediction-oriented true positives) and $1 - 0.5 = 0.5$ to b (false positives). For the top-right pixel (probability of 0.2), convection is *not* observed in the neighborhood, so this situation contributes 0.2 to b (false positives). Because there is no convection in the neighborhood, this situation contributes nothing to a_{pred} (prediction-oriented true positives).

function is pixelwise. However, these loss functions are computed after applying a spectral filter to the observations, thus removing convection at undesired scales, and training the U-net to predict convection at the desired scales. Hence, the filtered observations at one pixel incorporate information from the rest of the spatial domain, so the scale-separation loss functions computed thereafter are spatially enhanced.

In addition to the six scores in Table 1, we turn the Heidke score, Peirce score, and Gerrity score (Table 2) into scale-separation loss functions. We do not turn these scores into neighborhood loss functions, because their calculations involve true negatives. The neighborhood-based verification setting involves two types of true negatives—observation-oriented and prediction-oriented, like the two types of true positives—but unlike true positives, it is unclear from the literature how to use the two types of true negatives in computing verification scores. Because of this ambiguity and the fact that in a rare-event setting most true negatives are trivial (i.e., it easy to predict no convection in most cases), we do not use the Heidke, Peirce, or Gerrity scores in neighborhood loss functions.

The Heidke score is the number of correct predictions relative to a random model; the Peirce score is similar but for a random *and unbiased* model; and the Gerrity score is similar to the Heidke score except that the Gerrity score is equitable (gives random and constant models a score of 0, indicating no skill)

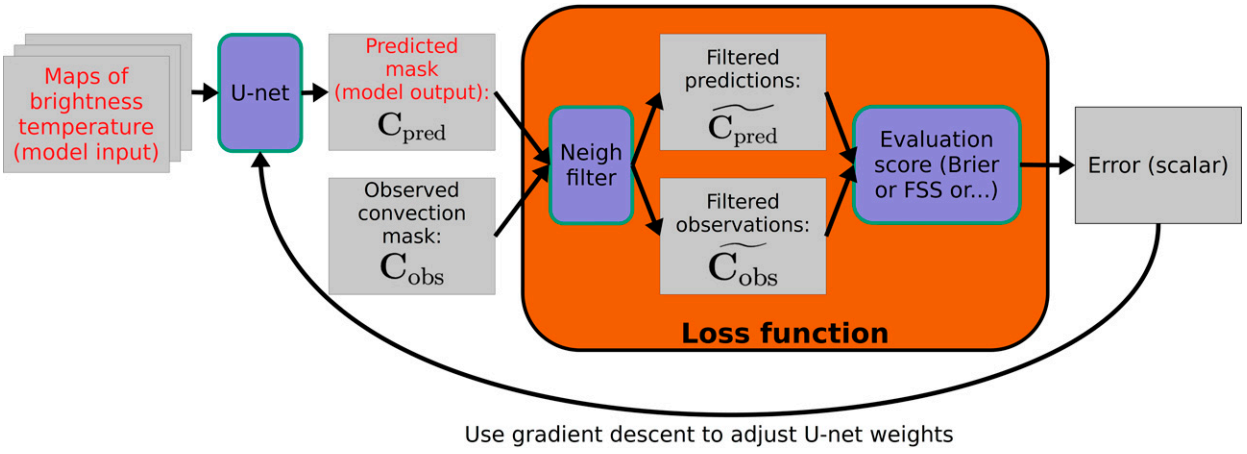

FIG. 6. Implementation of neighborhood filter inside a U-net. The neighborhood filter is embedded inside the loss function, where it is applied to both the predictions and the observations.

TABLE 2. Verification scores used only in scale-separation loss functions: a is the number of true positives, b is the number of false positives, c is the number of false negatives, d is the number of true negatives, and $N = a + b + c + d$ is the total number of examples. For each score, 0.0 indicates no skill and 1.0 is the optimal value. An intuitive definition of each score is provided in the main text.

Score	Definition	Range
Heidke score	$\frac{a + d - N_{\text{random}}}{N - N_{\text{random}}}$, where $N_{\text{random}} = [(a + b)(a + c) + (b + d)(c + d)]/N$	$(-\infty, 1]$
Peirce score	$\frac{a}{(a + c)} - \frac{b}{(b + d)}$ or $\frac{a}{(a + c)} + \frac{d}{(b + d)} - 1$	$[-1, 1]$
Gerrity score	$\frac{ar^{-1} + dr - b - c}{N}$, where r = event ratio $= N_{\text{events}}/N_{\text{nonevents}} = (a + c)/(b + d)$	$[-1, 1]$

and does not reward conservative models (i.e., those that err on the side of not predicting a rare event). Like the CSI, equations in Table 2 involve the contingency table, for which we use a probabilistic form rather than discretizing to create binary predictions. For example, consider a pixel with actual convection and a predicted convection probability of 0.8. This pixel contributes 0.8 to a (number of true positives) and $1 - 0.8 = 0.2$ to c (number of false negatives). As another example, consider a pixel with a predicted probability of 0.8 but no actual convection. This pixel contributes 0.8 to b (number of false positives) and $1 - 0.8 = 0.2$ to d (number of true negatives).

For each target field—that is, the observed convection mask at one time step—we use either Fourier or wavelet decomposition to implement the spectral filter. The two procedures are described schematically in Figs. 7 and 8 and in full detail in section 1 of the online supplemental material. The main advantage of Fourier decomposition over wavelet decomposition is richness of representation. In transforming

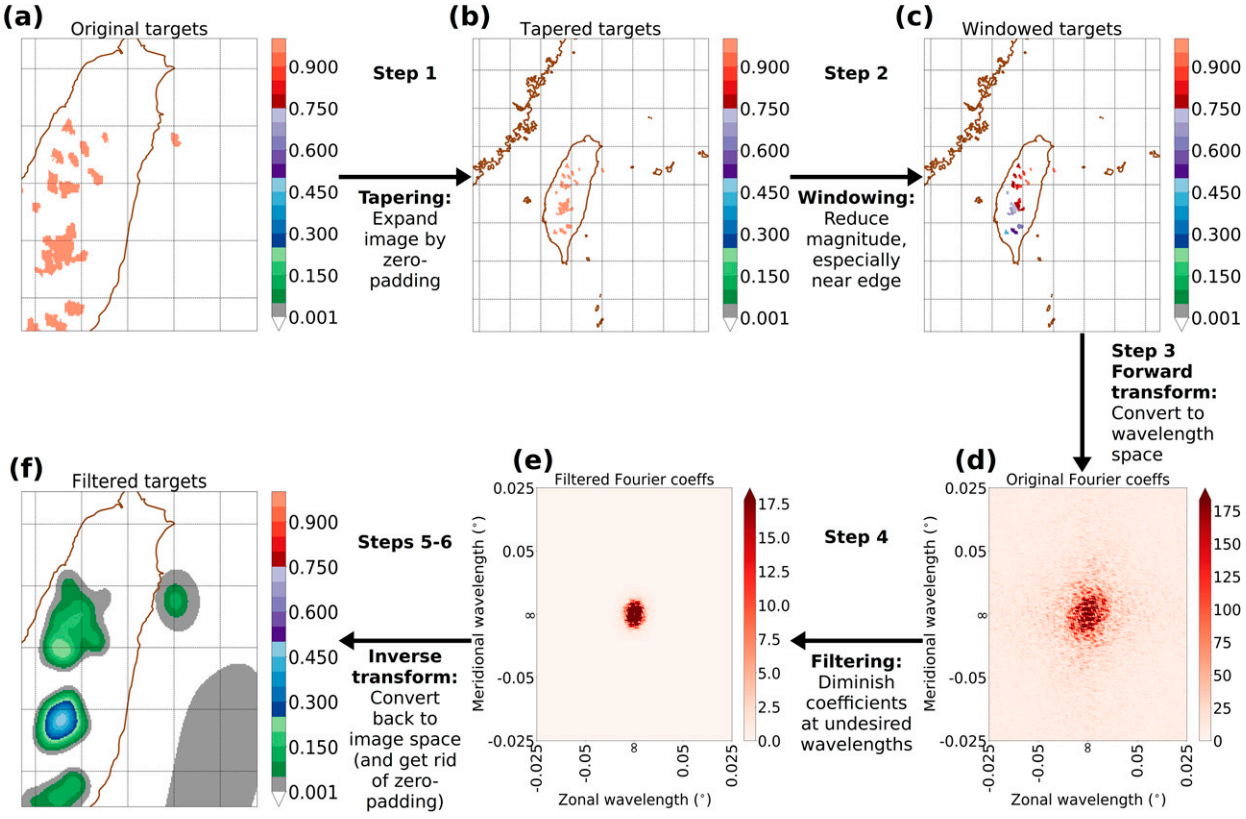

FIG. 7. Implementation of a spectral filter (in this case, allowing wavelengths from 0.5° to 2°) via Fourier decomposition. The original target field (i.e., convection mask) contains only 0s and 1s; after filtering, values range continuously from 0 to 1. For a detailed explanation of all steps in the procedure, see section 1 of the online supplemental material. (a) Original convection mask, with dimensions of 205 × 205. (b) Tapered convection mask, with dimensions of 615 × 615. (c) Tapered convection mask after applying Blackman–Harris window [Eq. (1) of the online supplemental material]. (d) Fourier spectrum (i.e., magnitude of each complex-valued coefficient), created by applying forward Fourier transform. (e) Fourier spectrum after applying Butterworth filter [supplemental Eq. (2)]. (f) Filtered convection mask, created by applying inverse Fourier transform and then removing the zero-padding.

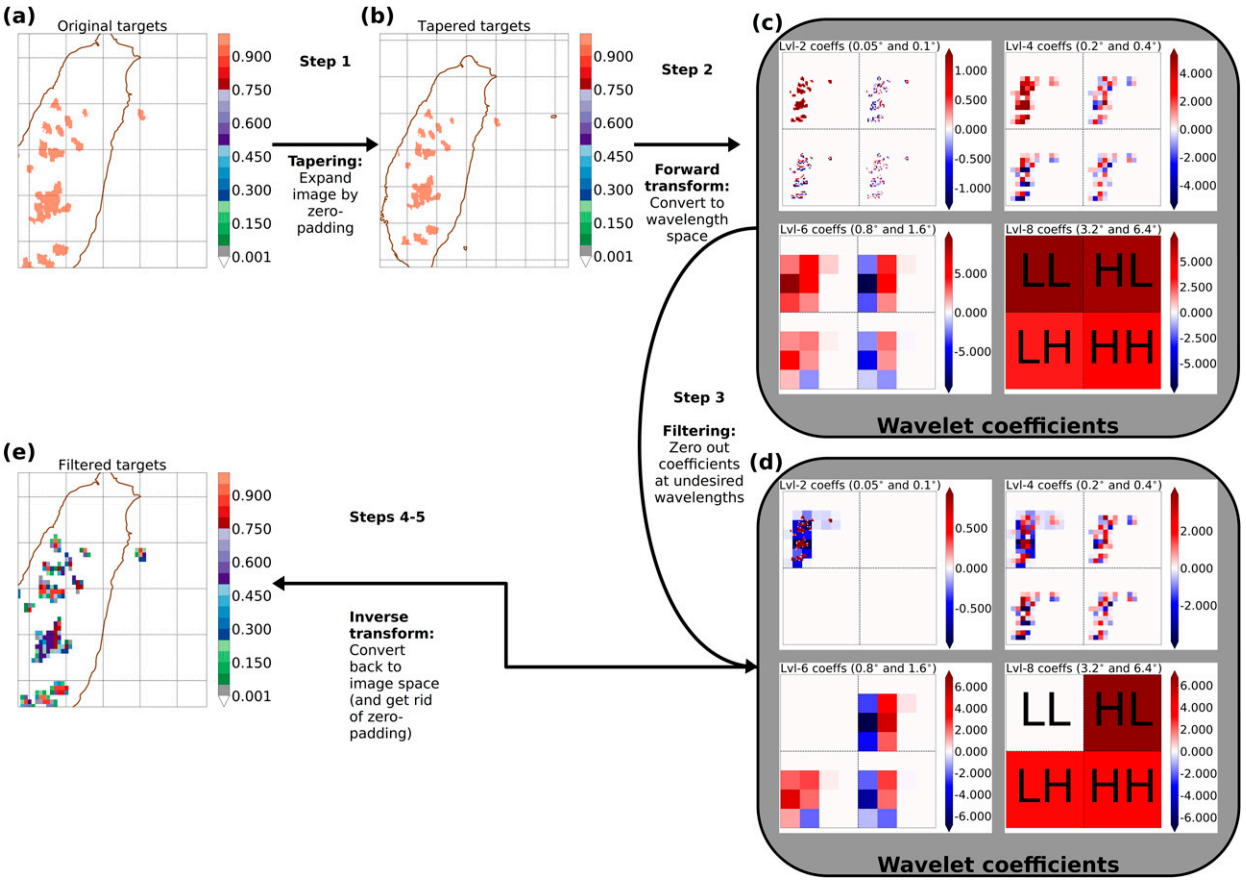

FIG. 8. Implementation of spectral filter (in this case, allowing wavelengths from 0.1° to 0.4°) via wavelet decomposition. The original target field (i.e., convection mask) contains only 0s and 1s; after filtering, values range continuously from 0 to 1. For a detailed explanation of all steps in the procedure, see section 1 of the online supplemental material. (a) Original field, with dimensions of 205 × 205. (b) Tapered field, with dimensions of 256 × 256. (c) Wavelet coefficients, created by applying forward wavelet transform. For brevity, only 4 of the 8 levels of decomposition are shown. Each subpanel here contains the LL, HL, LH, and HH coefficients (terms defined in the online supplemental material), as marked in the bottom-right subpanel. At level 2, the grid size is 64 × 64; the smaller wavelength represented (the "H" part) is 0.05°; and the larger wavelength represented (the "L" part) is 0.1°. In general, at level k, the grid size is $M2^{-k} \times N2^{-k}$, where M and N are the grid size of the original data; the smaller wavelength represented is $\delta 2^{k}$, where δ is the grid spacing of the original data; and the larger wavelength represented is $\delta 2^{k+1}$. (d) Wavelet coefficients after filtering. (e) Filtered field, created by applying inverse wavelet transform and then removing the zero-padding.

data from the spatial domain to the wavelength domain, for a spatial dimension with N pixels, Fourier decomposition represents N wavelengths while wavelet decomposition represents only $\log_2(N)$ wavelengths. The main advantage of wavelet decomposition is simplicity: it does not require windowing in the spatial domain (step 2 of Fourier procedure in supplemental section 1), and filtering out coefficients in the wavelength domain is much simpler (cf. step 4 of Fourier procedure and step 3 of wavelet procedure, both in supplemental section 1).

c. *Implementation*

Although neighborhood loss functions are easy to implement (section 3b), in general implementing custom loss functions comes with major challenges (Ebert-Uphoff et al. 2021). First, NNs are trained via gradient descent (section 6.5 of Goodfellow et al. 2016), so the loss function must be differentiable with respect to all NN weights. Second, the loss function must execute very quickly, since NN-training is already computationally expensive. Third, little guidance exists on writing custom loss functions, so researchers often encounter pitfalls.

First, we consider the implementation of loss functions with spectral filters using Fourier decomposition. The fast Fourier transform (FFT) and its inverse are predefined in many NN libraries. For example, in TensorFlow, the relevant methods are tensorflow.signal.fft2d and tensorflow.signal.ifft2d; both are differentiable. Other steps in Fourier decomposition, such as windowing and filtering (details of which are in supplemental section 1), can be cast as elementwise matrix multiplication, which is also differentiable. Thus, at first we implemented spectral filtering by

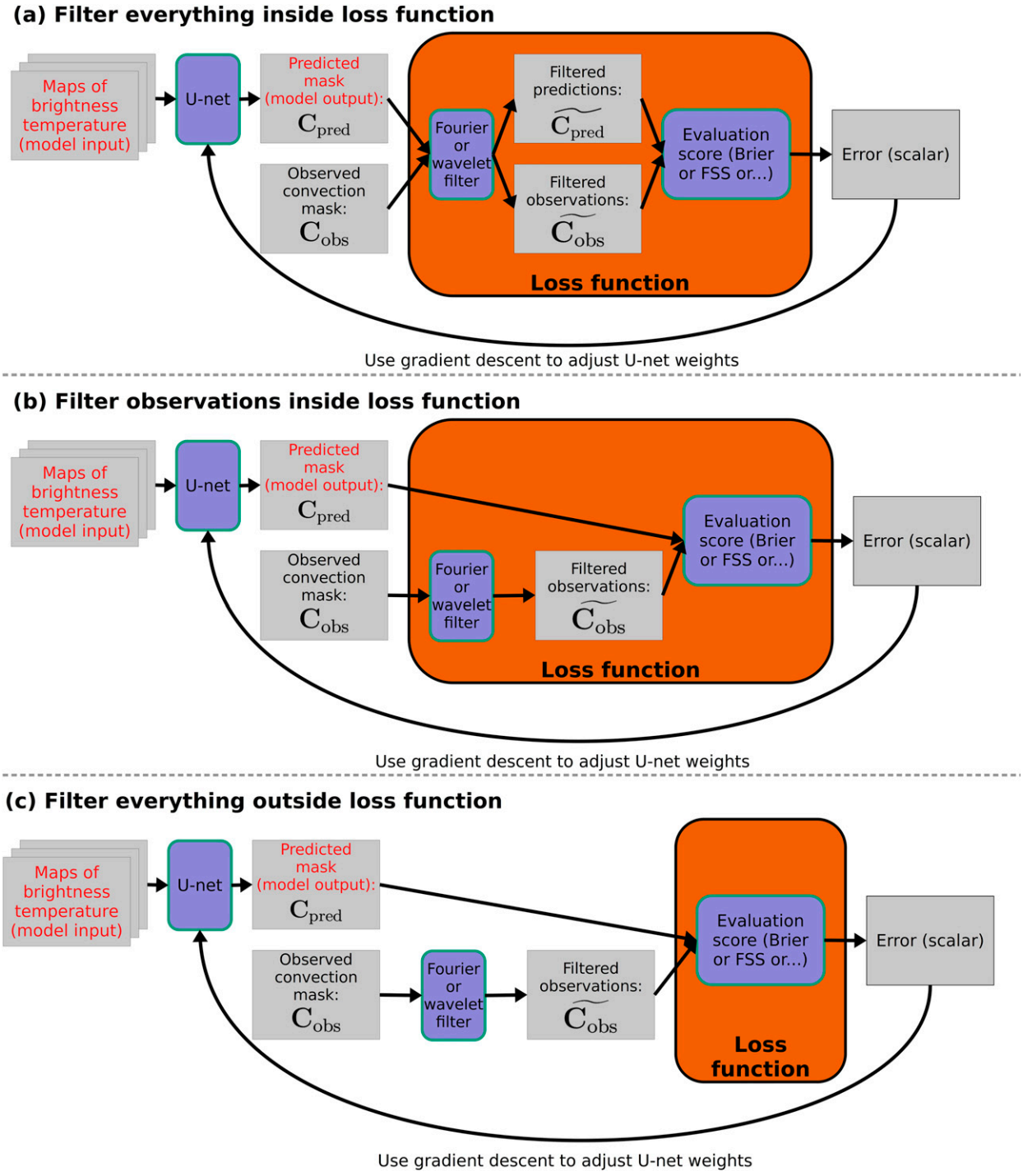

FIG. 9. Three ways to implement a spectral filter inside a U-net. (a) Filter everything (the predictions and observations) directly inside the loss function. (b) Filter only the observations, doing so inside the loss function. (c) Filter only the observations, doing so outside the loss function (i.e., as a preprocessing step). Methods (b) and (c) have the same effect, except that method c is less computationally expensive.

putting Fourier decomposition in the loss function, as shown in Fig. 9a. This is analogous to the schematic for neighborhood loss functions (Fig. 6), except that the neighborhood filter is replaced by a spectral filter using Fourier decomposition. In this setup, both the predictions and observations are filtered before they are compared with each other, but the NN's final output is the unfiltered predictions.

Experiments with this setup revealed a fundamental problem: the NN is free to produce arbitrary signals at the filtered wavelengths (i.e., those censored out by the spectral filter) with no penalty. Specifically, the resulting U-nets produced convection probabilities that varied erratically in space and time and bore little resemblance to the observations. Why did the same problem not occur with neighborhood loss functions? The spectral filters remove much more information from a spatial field than do the neighborhood filters, which merely smooth or dilate over a small window (Fig. 3), so neighborhood loss functions give the NN less freedom. However, if the filter size is large enough, neighborhood loss functions can lead to similar problems—see the discussion of "checkerboard patterns" in section 6a. Henceforth, we will call this problem—giving the NN too much freedom at filtered scales—the *excessive-freedom problem*.

One possible solution to the excessive-freedom problem is using a loss function with two terms: $w_{\text{filt}}s_{\text{filt}} + w_{\text{unfilt}}s_{\text{unfilt}}$, where s_{filt} is a score based on the filtered predictions and observations, s_{unfilt} is the same but for unfiltered predictions and observations, and the w_* are user-selected weights. However, this introduces more hyperparameters (the weights), which require careful tuning. Thus, we opted for a cleaner solution: filter the observations but not the predictions (Fig. 9b). This setup forces the NN to produce signals only at the desired wavelengths (i.e., those not removed by filtering), thus eliminating the excessive-freedom problem. However, if the spectral filter is applied only to the observations, it is completely equivalent—and more computationally efficient[4]—to apply the filter outside the loss function, as shown in Fig. 9c. This is the setup we actually use for spectral filtering via Fourier decomposition.

To implement loss functions with spectral filters using wavelet decomposition, we went through a similar thought process. For wavelet decomposition, the forward and inverse transforms are predefined operations in the WaveTF library (Versaci 2021), which interfaces with TensorFlow, and are also differentiable. However, filtering the wavelet coefficients (details of which are in section 1 of the online supplemental material) involves iterating over decomposition levels (typically done with a for loop) and deciding which coefficients to zero out (typically done with an if statement), which are hard to write in a differentiable form. Owing to these challenges and the excessive-freedom problem that arises with spectral filtering, we decided to apply wavelet-based filters outside the loss function, as with Fourier-based filters (Fig. 9c).

Note that the excessive-freedom problem can also occur for neighborhood loss functions with larger neighborhoods, as shown in section 6a.

[4] During training, each observed field $\mathbf{C}_{\text{obs}}$ is read from disk many times. Filtering inside the loss function means filtering $\mathbf{C}_{\text{obs}}$ every time it is read from disk; filtering outside the loss function means filtering $\mathbf{C}_{\text{obs}}$ once and storing the filtered field on disk.

TABLE 3. Loss functions used in the experiment. There are 48 neighborhood loss functions, yielded by combining 6 verification scores × 8 filter sizes. There are 288 scale-separation loss functions, yielded by combining 9 verification scores × 16 scale ranges × 2 spectral-filtering methods (Fourier and wavelet decomposition). Hence, there is a total of 48 + 288 = 336 loss functions, resulting in 336 U-net models. The same 336 loss functions are used as verification metrics post hoc, i.e., after training. The neighborhood loss functions are given as neighborhood half-width (pixels), and scale-separation loss functions are given as wavelength range (°).

Score	Loss functions
Neighborhood loss functions	
Brier score	0 (pixelwise), 1, 2, 3, 4, 6, 8, and 12
FSS	See above
IOU	See above
Dice coef	See above
CSI	See above
Cross entropy	See above
Scale-separation loss functions	
Brier score	≤0.025, 0.025–0.050, 0.050–0.100, 0.100–0.200, 0.200–0.400, 0.400–0.800, 0.800–1.600, ≥1.600, ≤0.1, ≤0.2, ≤0.4, ≤0.8, ≥0.1, ≥0.2, ≥0.4, and ≥0.8
FSS	See above
IOU	See above
Dice coef	See above
CSI	See above
Cross entropy	See above
Heidke score	See above
Peirce score	See above
Gerrity score	See above

5. Experiment

Combinations explored: We train many U-nets, each with a different loss function (Table 3), to predict convection at 1-h lead time. The goals of the experiment are to determine 1) the spatial characteristics of predictions (i.e., probability maps) produced by training with each loss function, 2) which loss functions lead to the best performance overall, and 3) which loss functions lead to the best performance in certain situations. As shown in Table 3, the total number of loss functions—and hence the total number of U-nets trained—is 336.

For training, following L21a and common practice in machine learning, we split the dataset into training (year 2016), validation (2017), and testing (2018). We use the training data to optimize model weights, the validation data to choose models for the final analysis (sections 6a, 6b), and the testing data for case studies in the final analysis (section 6c). For other details of the training (optimizer, number of epochs, etc.), see L21a.

For model verification after training (on the validation and testing data), we use a combination of visual (subjective) analysis and objective verification scores. In a study comparing the benefits of using different metrics as loss functions, it is nontrivial to decide which metric should be used for post hoc

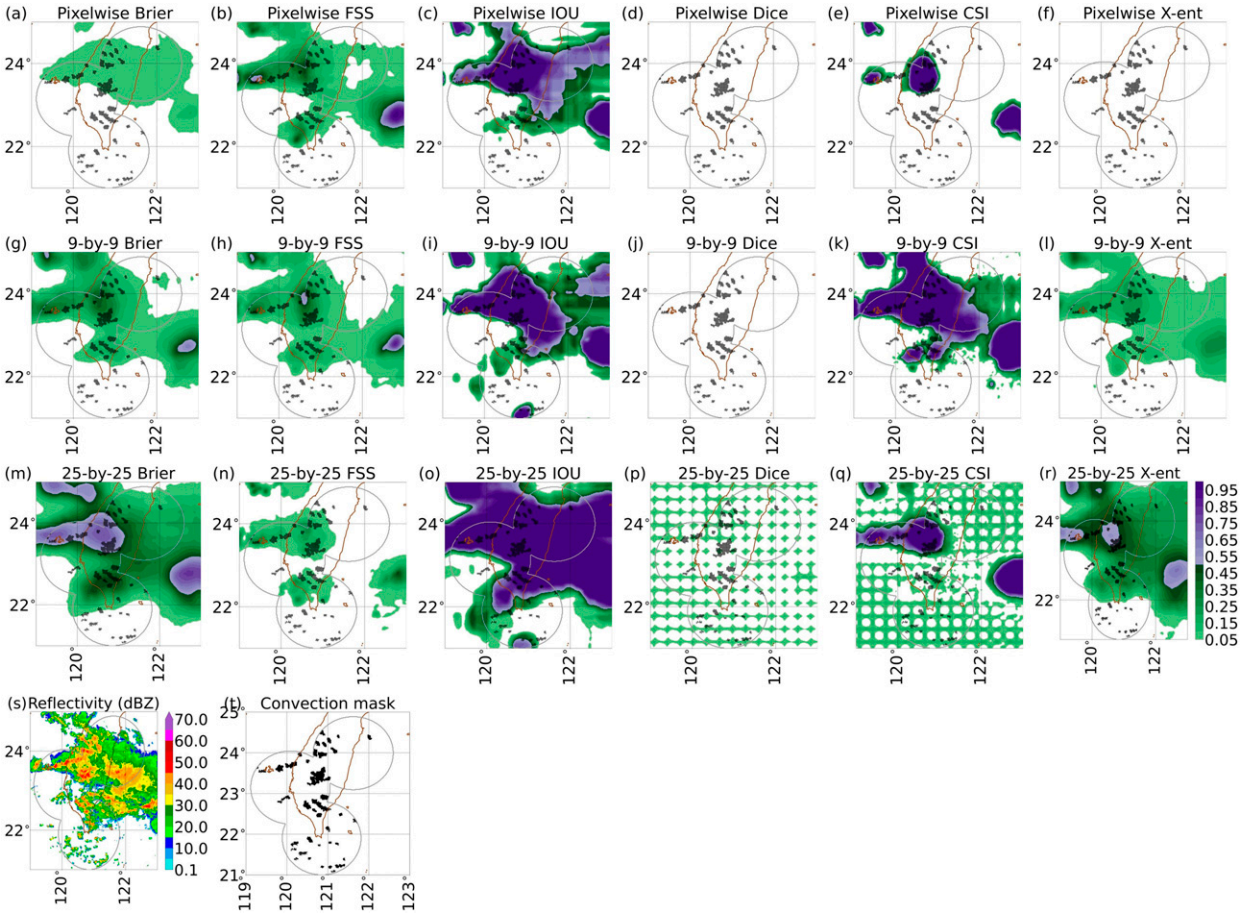

FIG. 10. Predictions valid at 2200 UTC 2 Jun 2017, made by U-nets trained with different neighborhood loss functions. Black dots show actual convection at 2200 UTC, according to the SL3D algorithm used to create targets. Black dots are not shown outside the 100-km range rings (gray-outlined circles), because targets (correct answers) here are unknown. Shown are convection probabilities forecast by U-nets trained to optimize scores with (a)–(f) a 1-by-1 neighborhood, i.e., pixelwise scores, (g)–(l) a 9-by-9 neighborhood, and (m)–(r) a 25-by-25 neighborhood. (s) Composite (column maximum) radar reflectivity valid at 2200 UTC. (t) Convection mask valid at 2200 UTC.

(after training) verification. (Henceforth, we use the word "loss function" for a verification method used during training and "metric" for the same verification method used after training.) Thus, we use all 336 loss functions as metrics for post hoc verification.

6. Results

Section 6a presents the preliminary analysis, in which all models are compared on a case study in the validation data. We use the preliminary analysis to dismiss all but 120 models as inappropriate for the convection application. Section 6b presents the intermediate analysis, where we use the 336 verification metrics to compare the 120 models remaining. We use the intermediate analysis to select models for the final analysis, presented in section 6c. In the final analysis, selected models are compared on several in-depth case studies in the testing data; we identify which models would be desirable for different risk thresholds and spatial scales of convection.

a. *Preliminary analysis*

Figure 10 shows predictions made by models trained with neighborhood loss functions, valid at 2200 UTC 2 June 2017, a randomly chosen time step in the validation data. Figure 11 is analogous but for scale-separation loss functions. Analyzing the various models' predictions yielded the following key results[5]:

- Training with the Dice coefficient (fourth column of Figs. 10 and 11) typically leads to very small probabilities (<0.05).

[5] Although Figs. 10 and 11 do not show all neighborhood or spectral filters, we have confirmed that for the convection application the following observations generalize to all filters. We have also confirmed that the following observations generalize to other cases and are not specific to 2200 UTC 2 June 2017.

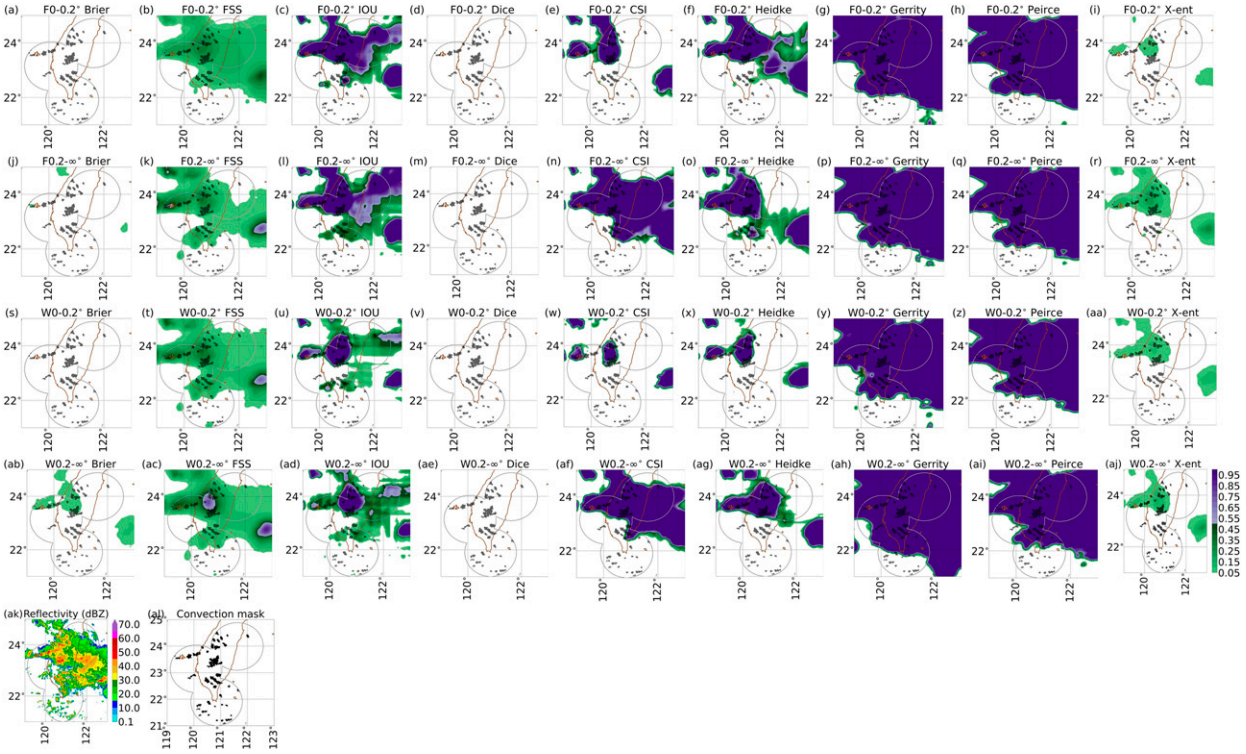

FIG. 11. Predictions valid at 2200 UTC 2 Jun 2017, made by U-nets trained with different scale-separation loss functions. Formatting is explained in the caption of Fig. 10. Shown are convection probabilities forecast by U-nets trained to optimize scores with (a)–(i) filter "F0°–0.2°," which uses Fourier decomposition to pass wavelengths of ≤0.2°, (j)–(r) filter "F0.2°–∞°," which uses Fourier decomposition to pass wavelengths ≥0.2°, (s)–(aa) filter "W0°–0.2°," which uses wavelet decomposition to pass wavelengths of ≤0.2°, and (ab)–(aj) filter "W0.2°–∞°," which uses wavelet decomposition to pass wavelengths ≥0.2°. (ak) Composite (column maximum) radar reflectivity valid at 2200 UTC. (al) Convection mask valid at 2200 UTC.

The Dice coefficient makes a poor loss function for rare events, because it rewards true negatives and true positives equally. Thus, for rare events, models can achieve a high Dice coefficient by always predicting negatives (no convection).

- Training with a 25-by-25 neighborhood leads to curious spatial patterns, such as the checkerboard patterns in Figs. 10p and 10q. These patterns are caused by the excessive-freedom problem (section 4c), which arises for large neighborhoods because the NN can produce erroneous small-scale patterns without penalty. When using neighborhood filters, one should keep the filter size small enough to avoid this effect.
- Training with the CSI (fifth column of Figs. 10 and 11) leads to large areas of very high probabilities (≥0.95), because the CSI rewards true positives but not true negatives. Interestingly, training with the CSI produces a very sharp model overall, with large areas of very low probabilities (<0.05) as well, including many false negatives on the edge of the convective area.
- Training with the IOU (third column of Figs. 10 and 11) and Heidke score (third-to-last column of Fig. 11) causes similar problems.
- Training with the Gerrity or Peirce score (last two columns of Fig. 11) also produces very high probabilities but, unlike the CSI and Heidke score, with fewer false negatives on the edge of the convective area.[6,7]
- For models trained with the Brier score, FSS, or cross entropy (first two columns and last column of Figs. 10 and 11), the aforementioned issues with the other loss functions are not present, except that models trained with a scale-separation Brier score produce quite low probabilities (<0.05 almost everywhere).

On the basis of the above observations, the rest of section 6 will focus solely on models trained with the Brier score, FSS, or cross entropy (120 of the 336 models).

b. *Intermediate analysis*

We use all 336 metrics for post hoc verification of the 120 remaining models—that is, those trained with the FSS,

[6] The Gerrity score has a well-known property of rewarding aggressive models (those that predict a rare event often), as documented in chapter 4 of Jolliffe and Stephenson (2012).

[7] The Peirce score is $[a/(a + c)] + [d/(b + d)] - 1$, with variables defined in the caption of Table 2; $b + d$ is the number of nonevents, and $a + c$ is the number of events, so $b + d \gg a + c$ for rare events. Thus, the second term has a much larger denominator, making the Peirce score more sensitive to the numerator of the first term (true positives) than that of the second term (true negatives).

Brier score, or cross entropy. The inputs to each metric are the unfiltered predictions and unfiltered observations ($\mathbf{C}_{\text{pred}}$ and $\mathbf{C}_{\text{obs}}$ in Figs. 6 and 9). Most metrics (all except the six pixelwise ones listed in Table 1) involve a spatial filter, in which case $\mathbf{C}_{\text{pred}}$ and $\mathbf{C}_{\text{obs}}$ are eventually filtered *inside the metric*.

Applying all metrics to all models would require the verification of 120 models using 336 metrics, yielding $336 \times 120 = 40\,320$ individual scores. One way to simplify the intermediate analysis would be to consider only a subset of the 336 metrics, that is, to prioritize some metrics over others. However, the purpose of this study is to explore which of the metrics result in the most promising loss functions—and thus the most promising models—not which metrics make the most appropriate final evaluation criteria. The latter question is heavily dependent on the needs of the end user. Thus, we prefer to include all metrics in the evaluation of the models for this exploratory study. To avoid information overload (i.e., considering 40 320 individual scores), we use the following procedure to generate for each model $\mathscr{M}$ a summary score for each spatial filter $\mathscr{F}$. This summary score can be interpreted as representing the general performance of $\mathscr{M}$ for the range of spatial scales highlighted by $\mathscr{F}$.

The purpose of this evaluation procedure is to identify those models that stand out in their performance for a particular scale and to study their properties further in the case studies in section 6c. Later studies with SELFs might take different approaches. In particular, if one knows for an application which metric best represents the goals of the end user, then one may use only that metric in model verification/selection. In contrast, here we want to explore any model that shows unusual abilities in any way for further study.

Procedure to generate summary score for each spatial filter $\mathscr{F}$ and model $\mathscr{M}$:

1) Compute all 336 metrics for $\mathscr{M}$.
2) Compute the rank on all 336 metrics for $\mathscr{M}$. For metrics involving a negatively oriented score—that is, the Brier score or cross entropy—the model with the lowest (highest) value receives a rank of 1 (120). For metrics involving a positively oriented score—all except the Brier score or cross entropy—the model with the lowest (highest) value receives a rank of 120 (1). The purpose of converting the individual scores to ranks is to allow us to take meaningful averages across different metrics (see step 3).
3) For each spatial filter $\mathscr{F}$, find all metrics using $\mathscr{F}$ and average the ranks over said metrics. This is denoted as the model's summary score for $\mathscr{F}$. The purpose of this step is to provide a single summary score for each model and each spatial filter.

Just like for the loss functions used during training, the metrics used during post hoc verification employ 40 different spatial filters, listed in Table 3. Thus, the above procedure yields 40 summary scores for each model. Finally, for each spatial filter used in the metrics, we identify the model with the best summary score. Results are summarized below and in Fig. 12.

- For 39 of 40 filters, the best model includes the FSS, but not the Brier score or cross entropy, in the loss function.
- For 20 of 40 filters, the best model is trained with pixelwise FSS.
- For 13 of 40 filters, the best model is trained with FSS after using wavelet decomposition to isolate wavelengths of 0.1°–∞°. Henceforth, we call this W0.1°–∞° FSS.
- For 1 of 40 filters, the best model is trained with F0.1°–∞° FSS, which is the same as W0.1°–∞° FSS but using Fourier decomposition.
- For 1 of 40 filters, the best model is trained with F0°–0.4° FSS.
- For 1 of 40 filters, the best model is trained with cross entropy using a 17-by-17 neighborhood filter.
- For the other 4 filters, the best model is trained with a neighborhood-based FSS. The most common neighborhood size for the best model (occurring 3 times) is 3×3.

Of the models listed above, we wish to select a small number for the final analysis, which uses in-depth case studies to understand how the models behave in different situations. There are two clear winners in the list, the first being the model trained with pixelwise FSS—that is, no spatial enhancement at all. The second clear winner in the list is the model trained with W0.1°–∞° FSS. The other scale-separation-based model in the list—trained with F0.1°–∞° FSS—has the best summary score for only one filter, but we still choose this model for final analysis, for symmetry with the W0.1°–∞° model. By comparing models trained with the same scales but different scale-separation methods (Fourier vs wavelet), we can see how these two methods affect model behavior. Furthermore, we choose the W0°–0.1° and F0°–0.1° models for final analysis. Although these models do not have the best summary score for any filter, the W0°–0.1° and F0°–0.1° filters used in their loss functions are complements to the filters used in the loss functions of the W0.1°–∞° and F0.1°–∞° models. This presents an opportunity to understand the differing effects of high- and low-pass filters on model behavior. Last, we choose the model trained with 17-by-17 FSS for final analysis. This model also does not have the best summary score for any filter, but including the pixelwise (1 by 1) and 17-by-17 FSS[8] presents an opportunity to understand the differing effects of small and large neighborhood filters on model behavior.

c. *Final analysis*

In this section we analyze the properties of forecasts produced by the six final models: those trained with pixelwise, 17-by-17, W0.1°–∞°, F0.1°–∞°, W0°–0.1°, and F0°–0.1° FSS. We consider the pixelwise model a baseline, since it is not spatially enhanced, against which to compare the other models, which are spatially enhanced. Specifically, we analyze case studies including convection at many scales: discrete storm cells,

[8] We would have chosen the 25-by-25 FSS for maximum contrast with 1-by-1 FSS, but models trained with 17-by-17 FSS perform substantially better than those trained with 25-by-25 FSS (e.g., Fig. 12).

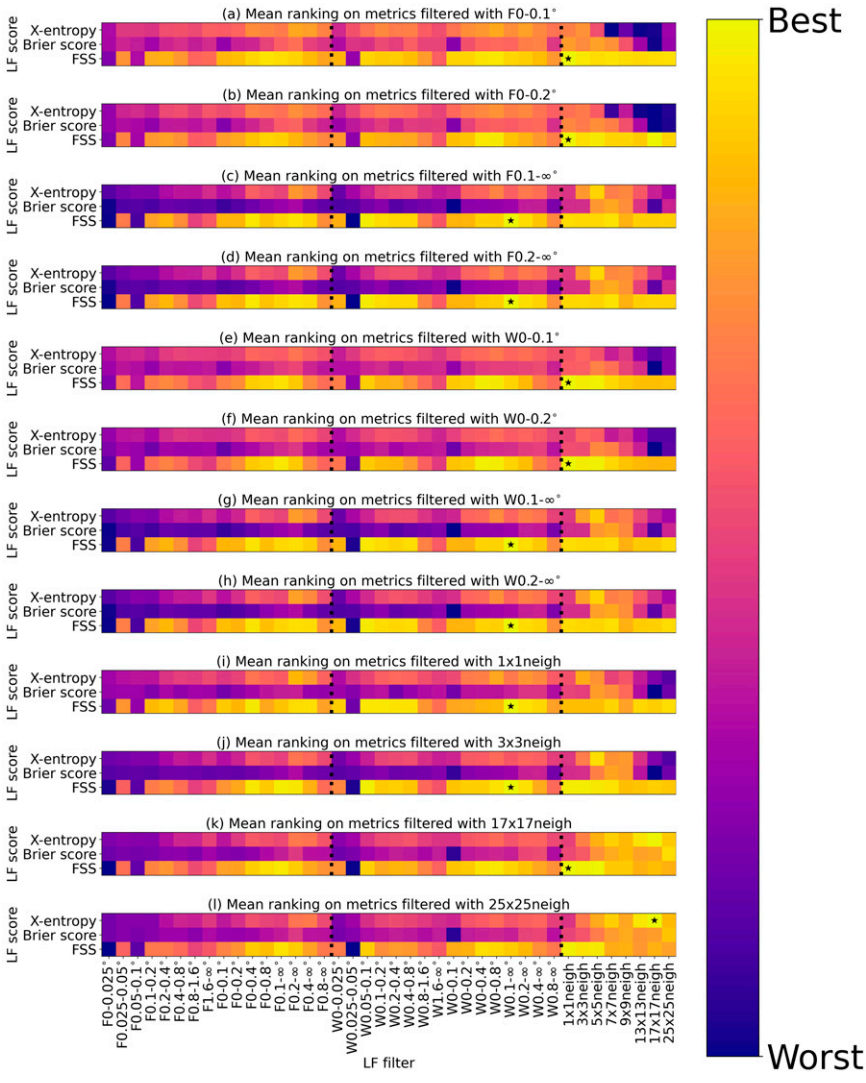

FIG. 12. Intermediate analysis. Each panel shows the summary score for 120 different models (those trained with FSS, Brier score, or cross entropy in the loss function) for verification metrics with one spatial filter, specified in the panel title. "LF score" on the *y* axis is the score used in the model's loss function, and "LF filter" on the *x* axis is the spatial filter used in the model's loss function—not to be confused with the filter used in the verification metric. In each panel, there are two dashed vertical lines, separating three groups of models: those trained with a Fourier-based filter (on the left), those trained with a wavelet-based filter (in the center), and those trained with a neighborhood filter (on the right). In each panel, the star denotes the best model on the given set of metrics. (a) Summary score on metrics using Fourier decomposition to highlight wavelengths of 0°–0.1°. (b)–(l) As in (a), but for metrics with different spatial filters. Only 12 sets of metrics (i.e., metrics involving only 12 of the 40 spatial filters) are shown, for the sake of brevity.

multicell clusters, a quasi-linear convective system (QLCS), and a tropical cyclone. The goal is to compare characteristics of the models' predictions in a qualitative manner so that we can subjectively evaluate the strengths and weaknesses of each loss function. For each time step shown in Figs. 13 and 14, we have been careful to highlight differences in model performance that are representative of the entire corresponding day (23 August and 3 June, respectively).

We preface the rest of this section with two notes on the way we interpret Figs. 13 and 14. First, we often use the terms "high risk threshold" and "low risk threshold." A user with a high risk threshold would prefer to be warned only of strong convection, while a user with a low risk threshold would prefer to be warned of any and all convection. Second, we consider forecast probabilities ≥0.05 to be "nonnegligible." Although 0.05 is a very low probability—and

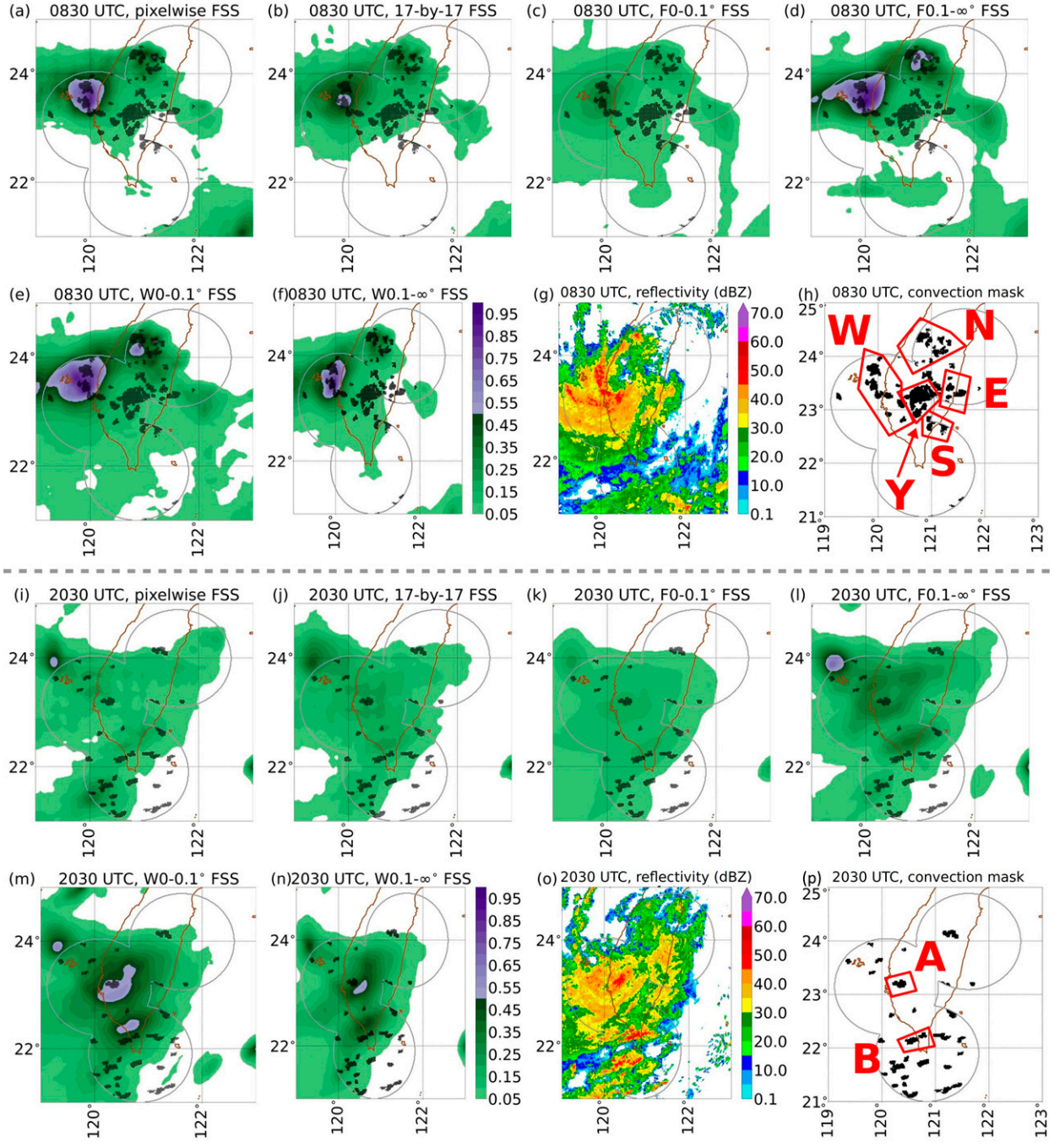

FIG. 13. Predictions made by final U-nets (each trained with a loss function involving a different spatial filter on FSS) for TD Luis. Formatting (black dots and panel titles) is explained in the captions of Figs. 10 and 11. (a)–(f) Forecast convection probabilities valid at 0830 UTC 23 Aug 2018. (g),(h) Composite reflectivity and convection mask, respectively, valid at 0830 UTC 23 Aug 2018. (i)–(n) Forecast convection probabilities valid at 2030 UTC 23 Aug 2018. (o),(p) Composite reflectivity and convection mask, respectively, valid at 2030 UTC 23 Aug 2018.

would be considered negligible in many applications—(i) our models rarely forecast higher probabilities[9] and (ii) in the case studies shown, the 0.05-probability contour corresponds well to the edge of the convective area.

Figure 13 shows two time steps on 23 August 2018, during the passage of Tropical Depression (TD) Luis. The first time step (0830 UTC; Figs. 13a–h) includes areas of strong convection (labeled "W" and "N") and weak convection (labeled

[9] For the six final models, the histogram of forecast probabilities is shown in Figs. 15a–f. Such a skewed distribution is common for rare-event problems.

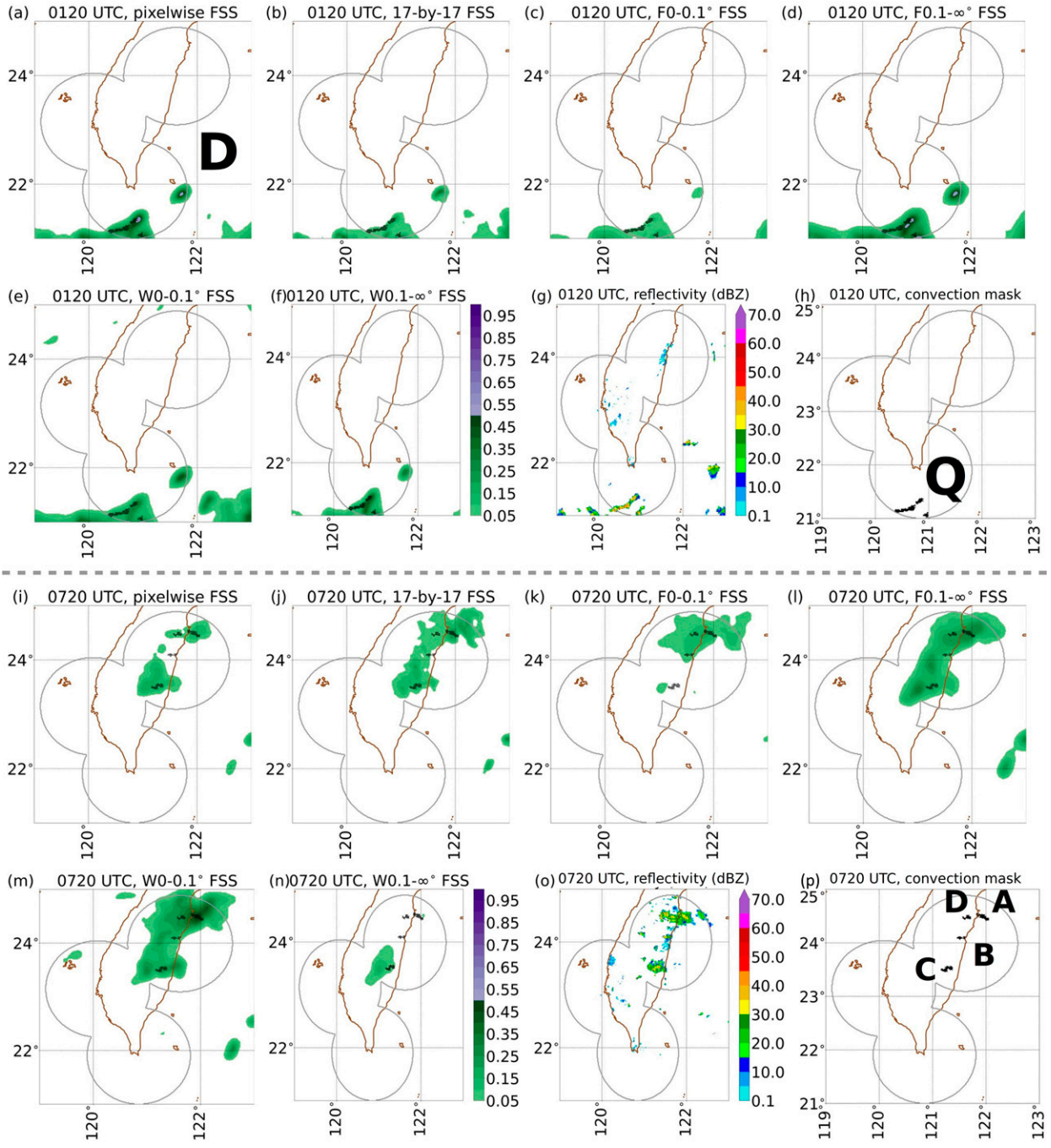

FIG. 14. Predictions made by final U-nets (each trained with a loss function involving a different spatial filter on FSS) for summer case. Formatting (black dots and panel titles) is explained in the captions of Figs. 10 and 11. (a)–(f) Forecast convection probabilities valid at 0120 UTC 3 Jun 2018. (g),(h) Composite reflectivity and convection mask, respectively, valid at 0120 UTC 3 Jun 2018. (i)–(n) Forecast convection probabilities valid at 0720 UTC 3 Jun 2018. (o),(p) Composite reflectivity and convection mask, respectively, valid at 0720 UTC 3 Jun 2018.

"E," "S," and "Y" for eye). The model trained with W0°–0.1° FSS (Fig. 13e) best captures the structure of the convective area, including both strong and weak convection, but it also produces the most false alarms—that is, area with probability ≥0.05 but no actual convection. Relative to the pixelwise model (Fig. 13a), the 17-by-17 model (Fig. 13b) has more extremely low probabilities and fewer extremely high probabilities. This is because the 17-by-17 filter smooths the target field (Fig. 3h), so

the 17-by-17 model is trained with fewer extremely high target values. Otherwise, outside area W, there are very slight differences between the pixelwise and 17-by-17 models. The F0.1°–∞° model (Fig. 13d) is similar to the pixelwise model but better at capturing convection in areas E and S. The W0°–0.1° model (Fig. 13e) is similar to the pixelwise model but better at capturing weak convection in general (areas E, S, and Y). The models not mentioned yet—W0.1°–∞° (Fig. 13f) and F0°–0.1° (Fig. 13c)—highlight the strong convection and produce small probabilities (<0.1) almost everywhere else. Between these two models, the W0.1°–∞° model produces much higher probabilities for the strong convection. Overall, for the first time step (0830 UTC) we judge that the W0°–0.1° model (Fig. 13e) would be best for a low risk threshold, while the W0.1°–∞° model (Fig. 13f) would be best for a high risk threshold.

At the second time step during TD Luis (2030 UTC; Figs. 13i–p), convection is weaker and more spread out. Two areas of strong convection are labeled A and B. Here, the main differences among models are (i) the F0.1°–∞° and W0°–0.1° models (Figs. 13l,m) are most aggressive, capturing the most convection but also producing the most false alarms; (ii) the W0°–0.1° model (Fig. 13m) captures slightly more convection than the F0.1°–∞° model (Fig. 13l) and produces substantially higher probabilities for strong convection in areas A and B; (iii) of the other four models, the W0.1°–∞° model (Fig. 13n) produces the highest probabilities for strong convection in areas A and B and has the fewest false alarms. Hence, for the second time step (2030 UTC) we judge that the W0°–0.1° model (Fig. 13m) would be best for a low risk threshold, while the W0.1°–∞° model (Fig. 13n) would be best for a high risk threshold.

Figure 14 shows a summer case (3 June 2018) featuring discrete storms, a QLCS at 0120 UTC, and multicell clusters at 0720 UTC. At 0120 UTC (Figs. 14a–h), where most of the convection is part of the QLCS (labeled "Q"), the pixelwise and F0.1°–∞° models (Figs. 14a,d) produce the highest probabilities, while the 17-by-17 and F0°–0.1° models (Figs. 14b,c) produce the lowest probabilities. Also, Fourier-based scale separation produces much more difference in probability magnitudes between the two scales (wavelengths of 0°–0.1° and 0.1°–∞°; Figs. 14c,d) than does wavelet-based scale separation (Figs. 14e,f). We believe that this is because, while Fourier and wavelet decomposition generally attribute the same amount of signal to the 0.1°–∞° scale (e.g., Figs. 4e,m), wavelet decomposition generally attributes more signal to the 0°–0.1° scale (e.g., Figs. 4,i). All models produce a false alarm (labeled "D") associated with convective decay, that is, the storm existed at forecast-issue time but has since died. At 0720 UTC (Figs. 14i–p), there are three multicell clusters (labeled "A"–"C") and a discrete storm ("D"). Clusters B and C are convective-initiation events, which only the 17-by-17, F0.1°–∞°, and W0°–0.1° models capture (Figs. 14j,l,m). In general, the W0.1°–∞° model (Fig. 14n) produces very low probabilities, likely because this model emphasizes larger scales and 0720 UTC features convection at smaller scales. For TD Luis, which featured convection at larger scales, the W0.1°–∞° model produced much higher probabilities (Figs. 14f,n). Only two models—F0.1°–∞° and W0°–0.1° (Figs. 14,l,m)—capture all convective pixels. Overall, for this day (3 June) we judge that the F0.1°–∞° or W0°–0.1° model (Figs. 14d,e,l,m) would be best for a low risk threshold. For a high risk threshold, it is difficult to judge which model would be best, since (i) all convection is quite weak, with reflectivity <45 dBZ; (ii) of the remaining models, no model captures all four convective areas at 0720 UTC and all models capture different convective areas.

Section 2 of the online supplemental material discusses a winter case, highlighting two important characteristics of all models. First, the models are poor at capturing weak isolated storms, especially for convective initiation. Second, the models have impressive sharpness for low probabilities—that is, at times with little to no convection, the models successfully produce very low probabilities throughout the entire domain. In other words, the models do not always produce swaths of elevated probabilities as in the summer case studies (Figs. 13 and 14).

Last, we have plotted two verification graphics: the attributes diagram (Hsu and Murphy 1986), which handles probabilistic forecasts, and the performance diagram (Roebber 2009), which handles deterministic forecasts. We summarize the quality of the performance diagram with the area under the curve (AUPD), which ranges over [0, 1] and is positively oriented. We summarize the quality of the attributes diagram with two numbers: reliability (REL), which ranges over [0, 1] and is negatively oriented, and Brier skill score (BSS), which ranges over $(-\infty, 1]$ and is positively oriented. See the original papers for more details. We have added consistency bars to the attributes diagram, following Bröcker and Smith (2007), using 100 bootstrapping iterations[10] and a 95% confidence level. Because of the discretization introduced by binning (i.e., points in the reliability curve are averaged over all cases with a probability of 0.00–0.05, 0.05–0.10, etc.), the reliability curve of a perfect model may deviate slightly from the 1-to-1 line. Consistency bars show how much deviation might be expected.

Figure 15 shows the attributes diagram and performance diagram for each final model—using pixelwise verification, which is the default for these diagrams—based on validation data only. In the attributes diagram (Figs. 15a–f), note that all models achieve a positive BSS, even those for which most of the reliability curve lies outside the positive-skill (i.e., positive BSS) area. This is because most examples fall in the lowest probability bin (as shown by the inset histogram), for which the corresponding (leftmost) point in the reliability curve is inside the positive-skill area. In other words, the lowest probabilities, which occur most often, are well calibrated even if the higher probabilities are not. Note that for rare events it is difficult to calibrate the higher probabilities (e.g., Fig. 5 of Gagne et al. 2015; Fig. 9 of Gagne et al. 2017; Fig. 10f of Lagerquist et al. 2017; Fig. 6 of Burke et al. 2020). In addition, Fig. 15 shows three interesting patterns. First, all models with a SELF (Figs. 15b–f) have a better attributes diagram than the model with a pixelwise loss function (Fig. 15a), in terms of both REL and BSS. For all models except F0.1°–∞°, the REL difference and BSS difference versus the pixelwise

[10] We tried 1000 bootstrapping iterations, but this would have taken too much computing time (>3 days).

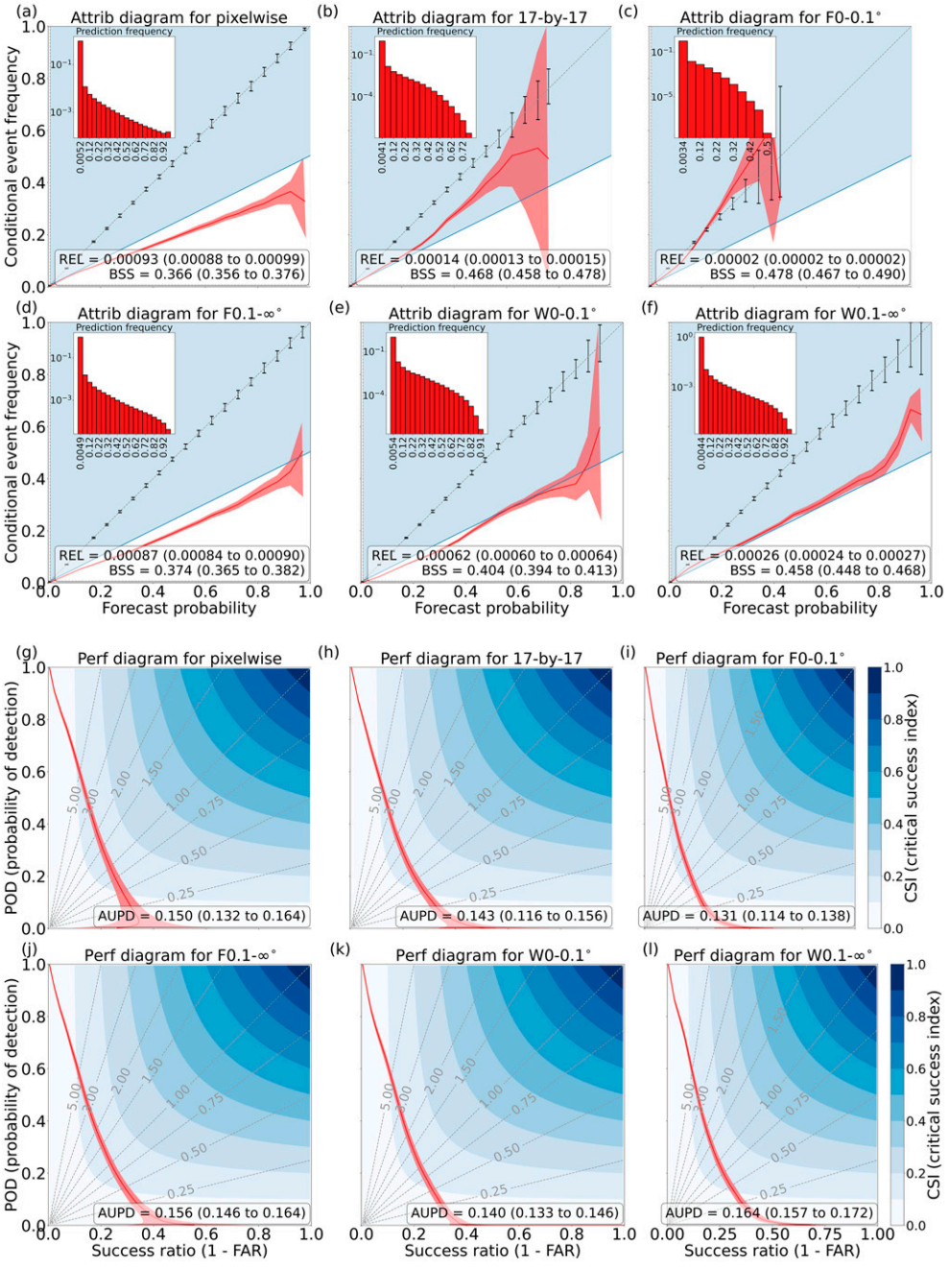

FIG. 15. Attributes diagram and performance diagram for each final model. Attributes diagram for model trained with (a) pixelwise FSS, (b) 17-by-17 FSS, (c) F0°–0.1° FSS, (d) F0.1°–∞° FSS, (e) W0°–0.1° FSS, and (f) W0.1°–∞° FSS. (g)–(l) As in (a)–(f), but for performance diagrams. The solid red lines show the mean, and the shaded red envelopes show the 95% confidence interval, based on bootstrapping 1000 times. In (a)–(f), the reliability curve (red) plots conditional event frequency vs forecast probability. The diagonal gray dashed line is the perfect-reliability line, error bars crossing the perfect-reliability line are consistency bars, the vertical gray dashed line is the climatology line, the horizontal gray dashed line is the no-resolution line, and the blue shading is the positive-skill area, where BSS > 0. REL and BSS are reported as the mean followed by 95% confidence interval. In (g)–(l), the red curve plots POD vs SR (both defined in Table 1). Each point corresponds to one probability threshold, used to convert probabilistic forecasts to deterministic. Dashed gray lines show the frequency bias (POD/SR; the optimal value is 1.0), and the blue color fill shows the CSI, defined in Table 1. AUPD is reported as the mean followed by 95% confidence interval.

model is significant at the 95% level, according to a two-sided paired bootstrapping test with 1000 iterations. Second, all models with a SELF (Figs. 15h–l) have a nearly equivalent performance diagram to the pixelwise model (Fig. 15g), in terms of AUPD (none of the five differences are significant at the 95% level). Third, in our judgement, the W0.1°–∞° model has the best combination of probabilistic forecasts (based on the attributes diagram) and deterministic forecasts (based on the performance diagram), with the best performance diagram of all six models and a reliability curve falling completely inside the positive-skill area. The W0.1°–∞° model seems preferable to all other models, including the pixelwise model. This is true despite the use of pixelwise verification in Fig. 15, which gives an edge to the pixelwise model. Thus, even in a traditional (pixelwise) verification setting, using SELFs can have benefit.

7. Discussion and future work

In this section we seek to answer the question from the title: can we integrate SV methods into NN loss functions for atmospheric science? We summarize our insights into this question below, while also considering the question of usefulness.

a. *Can we integrate neighborhood methods into loss functions? Is it useful?*

The answer to both questions is a clear yes. Neighborhood-based verification methods are easy to integrate into loss functions, and it is useful to train NNs to optimize the metrics that users care about, such as the well-established FSS. Note that we use a probabilistic version of the FSS—and all other verification scores—instead of thresholding to create binary forecasts. Although we omit thresholding mainly for technical reasons, we believe that thresholding, by destroying the information contained in continuous probabilistic forecasts, would degrade NN performance. Additionally, we learned the following lessons:

- Our sample application (nowcasting convection) does not *resoundingly* demonstrate the usefulness of neighborhood loss functions, as in many settings the best neighborhood loss function is a pixelwise one, using the trivial 1-by-1 neighborhood. We speculate that this is because the double penalty is not a major problem for our application, that is, 1-h-ahead forecasting of convection. However, this conclusion is moderated by three other observations. First, section 3 of the online supplemental material documents a similar experiment but for 2-h-ahead forecasting, where models trained with the 5-by-5, 9-by-9, and 13-by-13 neighborhood filters perform extremely well and much better than any model trained with a pixelwise loss function[11] (cf. Fig. 12 with Fig. S4 of the online supplemental material). Second, models trained with nontrivial neighborhood loss functions, such as the 17-by-17 FSS, produce much better-calibrated probabilities. Third, for applications where the double penalty is a major problem, such as detecting the locations of fronts (Justin et al. 2022) and wildfires (Earnest et al. 2022), neighborhood loss functions still appear to be crucial.
- We explored a much wider range of neighborhood loss functions than previous work (FSS only). We also found—at least for our application—that neighborhood loss functions including the FSS are better than those including other verification metrics. There is good reason to believe that this conclusion holds for predicting rare events in general.
- One should be careful when using large neighborhood sizes, as this may trigger the excessive-freedom problem, manifesting as checkerboard artifacts in the predictions.

b. *Can we integrate scale separation into loss functions? Is it useful?*

The answer here is more nuanced. Implementing a spectral filter in the loss function is difficult, and we needed to modify our approach to yield useful results, putting the spectral filter outside the loss function. However, the final approach still integrates scale separation into NN-training and shows different benefits than neighborhood loss functions. We learned the following lessons:

- It is possible to implement spectral filters with Fourier decomposition directly in the loss function. However, this leads to the excessive-freedom problem, as the NN can "go haywire" at scales removed by the spectral filter.
- It is technically challenging to implement spectral filters with wavelet decomposition directly in the loss function, and even if the technical (coding) challenges were overcome, this would lead to the excessive-freedom problem as well.
- The scale-separation approach has benefits not achieved by the neighborhood approach. First, in comparison with pixelwise model, the W0.1°–∞° model (trained to optimize FSS at wavelengths ≥0.1°) produces better probabilistic forecasts and comparable deterministic forecasts, even when assessed with pixelwise verification. Second—considering results from objective verification and the subjective case studies—the best models appear to be pixelwise and W0°–0.1° for a low risk threshold, W0.1°–∞° for a high risk threshold. Of these three models, two are trained with scale separation.

In addition to the technical details and experimental results summarized above, we provide Python code to implement SELFs. We believe that SELFs should be part of the developer's toolbox for training NNs—and other methods with loss functions, including other machine learning models and data assimilation—to optimize the performance measures that users truly care about. However, we have only scratched the surface in discovering what these tools can do and how best to use them. Future work will

[11] All three of these neighborhood-based models perform much better than that trained with the pixelwise FSS. However, this comparison is not very stringent, because the model trained with the pixelwise FSS failed to converge. As discussed in section 3 of the online supplemental material, in the interest of fairness and computing time, we do not retrain models that failed to converge.

proceed along three lines. First, we will develop temporally enhanced loss functions for predicting time series, which would ideally make predictions more temporally smooth (not varying erratically between time steps) and prevent the temporal version of the double penalty. Second, we will combine physical constraints (already used in loss functions for some geoscience applications) with SELFs. Third, we will investigate SELFs for more applications to build an understanding of which methods are best for which applications.

Acknowledgments. We thank the Taiwan CWB for providing the satellite and radar data used herein. We also thank an anonymous reviewer for very helpful comments, including the suggestion of additional experimentation. This work was partially supported by the NOAA Global Systems Laboratory, Cooperative Institute for Research in the Atmosphere, and NOAA Award NA19OAR4320073. Author Ebert-Uphoff's work was partially supported by NSF AI Institute Grant ICER-2019758 and NSF Grant OAC-1934668.

Data availability statement. Input data (satellite and radar images) are available upon request from the authors, as well as trained versions of all U-net models. We used version 2.0.0 of ML4convection (https://doi.org/10.5281/zenodo.6359915)—a Python library managed by author Lagerquist—to train and verify all models in this work. More important, we have created a Colab notebook (see version 1.0.0 here: https://github.com/thunderhoser/loss_function_paper_2022/blob/main/loss_functions_journal_paper_2022.ipynb) with all of the relevant code (independent of ML4convection), providing an end-to-end resource that people can use to experiment with our SELFs and implement their own.

REFERENCES

Ahmed, K., D. A. Sachindra, S. Shahid, M. C. Demirel, and E.-S. Chung, 2019: Selection of multi-model ensemble of general circulation models for the simulation of precipitation and maximum and minimum temperature based on spatial assessment metrics. *Hydrol. Earth Syst. Sci.*, **23**, 4803–4824, https://doi.org/10.5194/hess-23-4803-2019.

Bachmann, K., C. Keil, and M. Weissmann, 2018: Impact of radar data assimilation and orography on predictability of deep convection. *Quart. J. Roy. Meteor. Soc.*, **145**, 117–130, https://doi.org/10.1002/qj.3412.

Beucler, T., M. Pritchard, S. Rasp, J. Ott, P. Baldi, and P. Gentine, 2021: Enforcing analytic constraints in neural networks emulating physical systems. *Phys. Rev. Lett.*, **126**, 098302, https://doi.org/10.1103/PhysRevLett.126.098302.

Bröcker, J., and L. A. Smith, 2007: Increasing the reliability of reliability diagrams. *Wea. Forecasting*, **22**, 651–661, https://doi.org/10.1175/WAF993.1.

Brooks, H. E., 2004: Tornado-warning performance in the past and future: A perspective from signal detection theory. *Bull. Amer. Meteor. Soc.*, **85**, 837–843, https://doi.org/10.1175/BAMS-85-6-837.

Burke, A., N. Snook, D. J. Gagne II, S. McCorkle, and A. McGovern, 2020: Calibration of machine learning–based probabilistic hail predictions for operational forecasting. *Wea. Forecasting*, **35**, 149–168, https://doi.org/10.1175/WAF-D-19-0105.1.

Chen, Y., L. Bruzzone, L. Jiang, and Q. Sun, 2020: Aru-net: Reduction of atmospheric phase screen in SAR interferometry using attention-based deep residual U-net. *IEEE Trans. Geosci. Remote Sens.*, **59**, 5780–5793, https://doi.org/10.1109/TGRS.2020.3021765.

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

Earnest, B., A. McGovern, and I. L. Jirak, 2022: Using deep learning to predict the existence of wildfires with fuel data. *21st Conf. on Artificial Intelligence for Environmental Science*, Houston, TX, Amer. Meteor. Soc., 3.6, https://ams.confex.com/ams/102ANNUAL/meetingapp.cgi/Paper/395859.

Ebert-Uphoff, I., R. Lagerquist, K. Hilburn, Y. Lee, K. Haynes, J. Stock, C. Kumler, and J. Stewart, 2021: CIRA guide to custom loss functions for neural networks in environmental sciences—Version 1. arXiv, 2106.09757v1, https://doi.org/10.48550/arXiv.2106.09757.

Eigen, D., and R. Fergus, 2015: Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. *Proc. IEEE Int. Conf. on Computer Vision*, Santiago, Chile, Institute of Electrical and Electronics Engineers, 2650–2658, https://openaccess.thecvf.com/content_iccv_2015/html/Eigen_Predicting_Depth_Surface_ICCV_2015_paper.html.

Fujieda, S., K. Takayama, and T. Hachisuka, 2017: Wavelet convolutional neural networks for texture classification. arXiv, 1707.07394v1, https://doi.org/10.48550/arXiv.1707.07394.

Gagne, D. J., II, A. McGovern, J. Brotzge, M. Coniglio, J. Correia Jr., and M. Xue, 2015: Day-ahead hail prediction integrating machine learning with storm-scale numerical weather models. *Proc. Conf. Artificial Intelligence*, Austin, TX, Association for the Advancement of Artificial Intelligence, 3954–3960.

——, ——, S. E. Haupt, R. A. Sobash, J. K. Williams, and M. Xue, 2017: Storm-based probabilistic hail forecasting with machine learning applied to convection-allowing ensembles. *Wea. Forecasting*, **32**, 1819–1940, https://doi.org/10.1175/WAF-D-17-0010.1.

Gilleland, E., 2021: Novel measures for summarizing high-resolution forecast performance. *Adv. Stat. Climatol. Meteor. Oceanogr.*, **7**, 13–34, https://doi.org/10.5194/ascmo-7-13-2021.

——, D. Ahijevych, B. G. Brown, B. Casati, and E. Ebert, 2009: Intercomparison of spatial forecast verification methods. *Wea. Forecasting*, **24**, 1416–1430, https://doi.org/10.1175/2009WAF2222269.1.

Goodfellow, I. J., J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, 2014: Generative adversarial nets. *Conf. on Neural Information Processing Systems*, Montreal, QC, Canada, Neural Information Processing Systems Foundation, 1–9, https://papers.nips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf.

Goodfellow, I., Y. Bengio, and A. Courville, 2016: *Deep Learning.* MIT Press, 800 pp.

Gürsoy, Ö., and S. Engin, 2019: A wavelet neural network approach to predict daily river discharge using meteorological data. *Meas. Control*, **52**, 599–607, https://doi.org/10.1177/0020294019827977.

Hammernik, K., F. Knoll, D. Sodickson, and T. Pock, 2017: L2 or not L2: Impact of loss function design for deep learning MRI reconstruction. *Proc. Int. Society for Magnetic Resonance in Medicine*, Honolulu, HI, ISMRM, 0687, https://cds.ismrm.org/protected/17MProceedings/PDFfiles/0687.html.

Han, L., M. Chen, K. Chen, H. Chen, Y. Zhang, B. Lu, L. Song, and R. Qin, 2021: A deep learning method for bias correction of ECMWF 24–240 h forecasts. *Adv. Atmos. Sci.*, **38**, 1444–1459, https://doi.org/10.1007/s00376-021-0215-y.

Harder, P., and Coauthors, 2020: NightVision: Generating nighttime satellite imagery from infra-red observations. arXiv, 2011.07017v2, https://doi.org/10.48550/arXiv.2011.07017.

Heim, N., and J. Avery, 2019: Adaptive anomaly detection in chaotic time series with a spatially aware echo state network. arXiv, 1909.01709v1, https://doi.org/10.48550/arXiv.1909.01709.

Hsu, W.-R., 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.

Johnson, J., A. Alahi, and L. Fei-Fei, 2016: Perceptual losses for real-time style transfer and super-resolution. *European Conf. on Computer Vision*, Amsterdam, Netherlands, ECCV, 694–711, https://doi.org/10.1007/978-3-319-46475-6_43.

Jolliffe, I. T., and D. B. Stephenson, 2012: *Forecast Verification: A Practitioner's Guide in Atmospheric Science.* John Wiley and Sons, 296 pp.

Justin, A., C. Willingham, A. McGovern, and J. Allen, 2022: Toward operational real-time identification of frontal boundaries using machine learning: A 3D model. *Conf. on Artificial Intelligence for Environmental Science*, Houston, TX, Amer. Meteor. Soc., 3.3, https://ams.confex.com/ams/102ANNUAL/meetingapp.cgi/Paper/395669.

Karpatne, A., and Coauthors, 2017: Theory-guided data science: A new paradigm for scientific discovery from data. *IEEE Trans. Knowl. Data Eng.*, **29**, 2318–2331, https://doi.org/10.1109/TKDE.2017.2720168.

Kiruluta, A., 2017: Reducing deep network complexity with Fourier transform methods. arXiv, 1801.01451v2, https://doi.org/10.48550/arXiv.1801.01451.

Kumler-Bonfanti, C., J. Stewart, D. Hall, and M. Govett, 2020: Tropical and extratropical cyclone detection using deep learning. *J. Appl. Meteor. Climatol.*, **59**, 1971–1985, https://doi.org/10.1175/JAMC-D-20-0117.1.

Lagerquist, R., A. McGovern, and T. Smith, 2017: Machine learning for real-time prediction of damaging straight-line convective wind. *Wea. Forecasting*, **32**, 2175–2193, https://doi.org/10.1175/WAF-D-17-0038.1.

——, J. Stewart, I. Ebert-Uphoff, and C. Kumler, 2021a: Using deep learning to nowcast the spatial coverage of convection from *Himawari-8* satellite data. *Mon. Wea. Rev.*, **149**, 3897–3921, https://doi.org/10.1175/MWR-D-21-0096.1.

——, D. Turner, I. Ebert-Uphoff, J. Stewart, and V. Hagerty, 2021b: Using deep learning to emulate and accelerate a radiative transfer model. *J. Atmos. Oceanic Technol.*, **38**, 1673–1696, https://doi.org/10.1175/JTECH-D-21-0007.1.

Ledig, C., and Coauthors, 2017: Photo-realistic single image super-resolution using a generative adversarial network. *Proc. IEEE. Conf. on Computer Vision and Pattern Recognition*, Honolulu, HI, Institute of Electrical and Electronics Engineers, 4681–4690, https://openaccess.thecvf.com/content_cvpr_2017/html/Ledig_Photo-Realistic_Single_Image_CVPR_2017_paper.html.

Lee-Thorp, J., J. Ainslie, I. Eckstein, and S. Ontañón, 2021: Fnet: Mixing tokens with Fourier transforms. arXiv, 2105.03824v4, https://doi.org/10.48550/arXiv.2105.03824.

Li, Q., L. Shen, S. Guo, and Z. Lai, 2020a: Wavelet integrated CNNS for noise-robust image classification. *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, Seattle, WA, Institute of Electrical and Electronics Engineers, 7245–7254, https://openaccess.thecvf.com/content_CVPR_2020/html/Li_Wavelet_Integrated_CNNs_for_Noise-Robust_Image_Classification_CVPR_2020_paper.html.

Li, Z., N. Kovachki, K. Azizzadenesheli, B. Liu, A. Stuart, K. Bhattacharya, and A. Anandkumar, 2020b: Multipole graph neural operator for parametric partial differential equations. Proc. *Advances in Neural Information Processing Systems 33 (NeurIPS 2020)*, Online, Neural Information Processing Systems, 6755–6766, https://proceedings.neurips.cc/paper/2020/hash/4b21cf96d4cf612f239a6c322b10c8fe-Abstract.html.

——, ——, ——, ——, K. Bhattacharya, A. Stuart, and A. Anandkumar, 2021: Fourier neural operator for parametric partial differential equations. *Proc. Int. Conf. on Learning Representations*, Online, ICLR, https://iclr.cc/virtual/2021/poster/3281.

Loken, E. D., A. J. Clark, M. Xue, and F. Kong, 2019: Spread and skill in mixed- and single-physics convection-allowing ensembles. *Wea. Forecasting*, **34**, 305–330, https://doi.org/10.1175/WAF-D-18-0078.1.

Lu, L., P. Jin, and G. Karniadakis, 2019: Deeponet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. arXiv, 1910.03193v3, https://doi.org/10.48550/arXiv.1910.03193.

Maas, A. L., A. Y. Hannun, and A. Y. Ng, 2013: Rectifier nonlinearities improve neural network acoustic models. *Proc. Int. Conf. on Machine Learning*, Atlanta, GA, International Machine Learning Society, 3, http://robotics.stanford.edu/~amaas/papers/relu_hybrid_icml2013_final.pdf.

Mathieu, M., M. Henaff, and Y. LeCun, 2014: Fast training of convolutional networks through FFTs. arXiv, 1312.5851v5, https://doi.org/10.48550/arXiv.1312.5851.

Mittermaier, M., N. Roberts, and S. A. Thompson, 2013: A long-term assessment of precipitation forecast skill using the fractions skill score. *Meteor. Appl.*, **20**, 176–186, https://doi.org/10.1002/met.296.

Partal, T., H. K. Cigizoglu, and E. Kahya, 2015: Daily precipitation predictions using three different wavelet neural network algorithms by meteorological data. *Stochastic Environ. Res. Risk Assess.*, **29**, 1317–1329, https://doi.org/10.1007/s00477-015-1061-1.

Pathak, J., and Coauthors, 2022: FourCastNet: A global data-driven high-resolution weather model using adaptive Fourier neural operators. arXiv, 2202.11214v1, https://doi.org/10.48550/arXiv.2202.11214.

Pratt, H., B. Williams, F. Coenen, and Y. Zheng, 2017: FCNN: Fourier convolutional neural networks. *Machine Learning and Knowledge Discovery in Databases*, M. Ceci et al., Eds., Lecture Notes in Computer Science Series, Vol. 10534, Springer, 786–798, https://doi.org/10.1007/978-3-319-71249-9_47.

Qian, X., and H. Wang, 2021: Evaluation of different storm parameters as the proxies for gridded total lightning flash rates: A convection-allowing model study. *Atmosphere*, **12**, 95, https://doi.org/10.3390/atmos12010095.

Rippel, O., J. Snoek, and R. Adams, 2015: Spectral representations for convolutional neural networks. *Proc. 28th Int. Conf. on Neural Information Processing Systems*, 2449–2457, Montreal, QC, Canada, Association for Computing Machinery, https://dl.acm.org/doi/10.5555/2969442.2969513.

Roberts, N. M., and H. W. Lean, 2008: Scale-selective verification of rainfall accumulations from high-resolution forecasts of convective events. *Mon. Wea. Rev.*, **136**, 78–97, https://doi.org/10.1175/2007MWR2123.1.

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

Ronneberger, O., P. Fischer, and T. Brox, 2015: U-net: Convolutional networks for biomedical image segmentation. *Int. Conf. on Medical Image Computing and Computer-assisted Intervention*, Munich, Germany, Technical University of Munich, 234–241, https://doi.org/10.1007/978-3-319-24574-4_28.

Sadeghi, M., P. Nguyen, K. Hsu, and S. Sorooshian, 2020: Improving near real-time precipitation estimation using a U-Net convolutional neural network and geographical information. *Environ. Modell. Software*, **134**, 104856, https://doi.org/10.1016/j.envsoft.2020.104856.

Sha, Y., D. J. Gagne II, G. West, and R. Stull, 2020a: Deep-learning-based gridded downscaling of surface meteorological variables in complex terrain. Part I: Daily maximum and minimum 2-m temperature. *J. Appl. Meteor. Climatol.*, **59**, 2057–2073, https://doi.org/10.1175/JAMC-D-20-0057.1.

——, ——, ——, and ——, 2020b: Deep-learning-based gridded downscaling of surface meteorological variables in complex terrain. Part II: Daily precipitation. *J. Appl. Meteor. Climatol.*, **59**, 2075–2092, https://doi.org/10.1175/JAMC-D-20-0058.1.

Snell, J., K. Ridgeway, R. Liao, B. D. Roads, M. C. Mozer, and R. S. Zemel, 2017: Learning to generate images with perceptual similarity metrics. *Proc. Int. Conf. on Image Processing*, Beijing, China, Institute of Electrical and Electronics Engineers, 4277–4281, https://doi.org/10.1109/ICIP.2017.8297089.

Sobash, R. A., J. S. Kain, D. R. Bright, A. R. Dean, M. C. Coniglio, and S. J. Weiss, 2011: Probabilistic forecast guidance for severe thunderstorms based on the identification of extreme phenomena in convection-allowing model forecasts. *Wea. Forecasting*, **26**, 714–728, https://doi.org/10.1175/WAF-D-10-05046.1.

Starzec, M., C. R. Hometer, and G. L. Mullendore, 2017: Storm labeling in three dimensions (SL3D): A volumetric radar echo and dual-polarization updraft classification algorithm. *Mon. Wea. Rev.*, **145**, 1127–1145, https://doi.org/10.1175/MWR-D-16-0089.1.

Stengel, K., A. Glaws, D. Hettinger, and R. N. King, 2020: Adversarial super-resolution of climatological wind and solar data. *Proc. Natl. Acad. Sci. USA*, **117**, 16 805–16 815, https://doi.org/10.1073/pnas.1918964117.

Versaci, F., 2021: WaveTF: A fast 2D wavelet transform for machine learning in Keras. *Proc. Int. Conf. Pattern Recognition*, Online, ICPR, 605–618, https://doi.org/10.1007/978-3-030-68763-2_46.

Wang, C., C. Xu, C. Wang, and D. Tao, 2018: Perceptual adversarial networks for image-to-image transformation. *IEEE Trans. Image Process.*, **27**, 4066–4079, https://doi.org/10.1109/TIP.2018.2836316.

Wang, L., Y. Zhang, and J. Feng, 2005: On the Euclidean distance of images. *IEEE Trans. Pattern Anal. Mach. Intell.*, **27**, 1334–1339, https://doi.org/10.1109/TPAMI.2005.165.

Wang, Z., and A. C. Bovik, 2009: Mean squared error: Love it or leave it? A new look at signal fidelity measures. *IEEE Signal Process. Mag.*, **26**, 98–117, https://doi.org/10.1109/MSP.2008.930649.

——, ——, H. R. Sheikh, and E. P. Simoncelli, 2004: Image quality assessment: From error visibility to structural similarity. *IEEE Trans. Image Process.*, **13**, 600–612, https://doi.org/10.1109/TIP.2003.819861.

Weusthoff, T., F. Ament, M. Arpagaus, and M. W. Rotach, 2010: Assessing the benefits of convection-permitting models by neighborhood verification: Examples from MAP D-PHASE. *Mon. Wea. Rev.*, **138**, 3418–3433, https://doi.org/10.1175/2010MWR3380.1.

Willard, J., X. Jia, S. Xu, M. Steinbach, and V. Kumar, 2020: Integrating physics-based modeling with machine learning: A survey. arXiv, 2003.04919v6, https://doi.org/10.48550/arXiv.2003.04919.

Zhang, J., P. F. Craigmile, and N. Cressie, 2008: Loss function approaches to predict a spatial quantile and its exceedance region. *Technometrics*, **50**, 216–227, https://doi.org/10.1198/004017008000000226.

Zhao, H., O. Gallo, I. Frosio, and J. Kautz, 2017: Loss functions for image restoration with neural networks. *IEEE Trans. Comput. Imaging*, **3**, 47–57, https://doi.org/10.1109/TCI.2016.2644865.