Creating and evaluating uncertainty estimates with neural networks for environmental-science applications

Ryan Lagerquist[1], Katherine Haynes[1], Marie McGraw[1], Kate Musgrave[1], and Imme Ebert-Uphoff[1]

[1]Cooperative Institute for Research in the Atmosphere

November 23, 2022

Abstract

Neural networks (NNs) have become an important tool for prediction and classification tasks in environmental science applications. Since many such tasks inform life-and-death decision or policy making, it is crucial to not only provide predictions but also gain an understanding of the uncertainty of these predictions. Until recently there were very few tools available to provide uncertainty quantification (UQ) estimates for NN predictions, but over the last two years the computer science field has developed numerous new methods for this purpose, and many research groups are exploring how to put these methods into practice for environmental science applications. In this work we provide a brief, accessible introduction to four of these UQ methods, then focus on tools for the next step, namely to answer the question: Once we obtain an uncertainty estimate (using any method), how do we know whether it is good? To answer this, we highlight four different evaluation methods that are particularly suitable to evaluate NN uncertainty estimates for environmental science applications. We demonstrate the UQ evaluation methods for two real-world problems: (1) estimating vertical profiles of atmospheric dewpoint (regression task) and (2) predicting convection over Taiwan based on Himawari-8 satellite imagery (classification task). We also provide accompanying Jupyter notebooks with Python code for implementing the uncertainty estimation and UQ evaluation methods discussed herein. This article provides the environmental-science community with the knowledge and tools to start incorporating the large number of emerging UQ methods into their research.

Creating and evaluating uncertainty estimates with neural networks for environmental-science applications

This article has been submitted to the AMS journal *Artificial Intelligence for the Earth Systems*.

Katherine Haynes[a] , Ryan Lagerquist[a,b] , Marie McGraw[a] , Kate Musgrave[a] , Imme Ebert-Uphoff[a,c]

[a] *Cooperative Institute for Research in the Atmosphere (CIRA), Colorado State University, Fort Collins, CO*

[b] *National Oceanic and Atmospheric Administration (NOAA) Earth System Research Laboratory (ESRL) / Global Systems Laboratory (GSL), Boulder, Colorado*

[c] *Department of Electrical and Computer Engineering, Colorado State University, Fort Collins, Colorado*

Corresponding author: Katherine Haynes, katherine.haynes@colostate.edu

ABSTRACT: Neural networks (NNs) have become an important tool for prediction and classification tasks in environmental science applications. Since many such tasks inform life-and-death decision or policy making, it is crucial to not only provide predictions but also gain an understanding of the uncertainty of these predictions. Until recently there were very few tools available to provide uncertainty quantification (UQ) estimates for NN predictions, but over the last two years the computer science field has developed numerous new methods for this purpose, and many research groups are exploring how to put these methods into practice for environmental science applications. In this work we provide a brief, accessible introduction to four of these UQ methods, then focus on tools for the next step, namely to answer the question: *Once we obtain an uncertainty estimate (using any method), how do we know whether it is good?* To answer this, we highlight four different evaluation methods that are particularly suitable to evaluate NN uncertainty estimates for environmental science applications. We demonstrate the UQ evaluation methods for two real-world problems: (1) estimating vertical profiles of atmospheric dewpoint (regression task) and (2) predicting convection over Taiwan based on Himawari-8 satellite imagery (classification task). We also provide accompanying Jupyter notebooks with Python code for implementing the uncertainty estimation and UQ evaluation methods discussed herein. This article provides the environmental-science community with the knowledge and tools to start incorporating the large number of emerging UQ methods into their research.

SIGNIFICANCE STATEMENT: AI methods are used in many meteorological applications, including for life-and-death decision making. New methods have been developed in recent years in computer science to provide urgently needed uncertainty estimates for such methods. We seek to accelerate adoption of these methods in the environmental science community with an accessible introduction to 1) simple methods for calculating uncertainty estimates of AI methods, and 2) methods for evaluating such estimates for environmental science applications.

1. Introduction

Neural networks (NNs) have become widely used tools for both prediction and classification tasks in the environmental sciences. Many environmental science tasks (*e.g.*, predicting ocean wave heights, forecasting severe weather) are used in critical decision and policy making. These decision makers need not only a prediction of an outcome, but also the uncertainty surrounding that prediction.

Previously, tools to provide uncertainty quantification (UQ) estimates for NN predictions have been scarcely available; however, over the last two years the field of computer science has seen a surge in this area. This article aims to aid the environmental science community in applying these tools.

To do this, we first provide background on what uncertainties the environmental sciences are aiming to estimate in machine learning (ML) models (Section 2). Next,

we provide a brief, accessible introduction to four types of these UQ methods (Section 3):

1. Probabilistic Prediction Using Continuous Ranked Probability Score (CRPS),
2. Parametric Regression,
3. Quantile Regression, and
4. Monte Carlo Dropout (and pointers for Bayesian Neural networks).

In Section 4 we discuss tools for the next step, namely to answer the question: *Once we obtain an uncertainty estimate using any method, how do we evaluate the quality of this estimate for our application?* We highlight four different evaluation methods that are particularly suitable to evaluate NN uncertainty estimates for environmental science applications. These are:

1. Attributes diagram,
2. Spread-skill plot,
3. Probability integral transform (PIT) histogram, and
4. Discard test.

In Sections 5 and 6 we demonstrate how to use the UQ evaluation methods for two real-world problems:

- Estimating vertical profiles of atmospheric dewpoint (regression task);
- Predicting convection over Taiwan based on Himawari-8 satellite imagery (pixel-wise classification task).

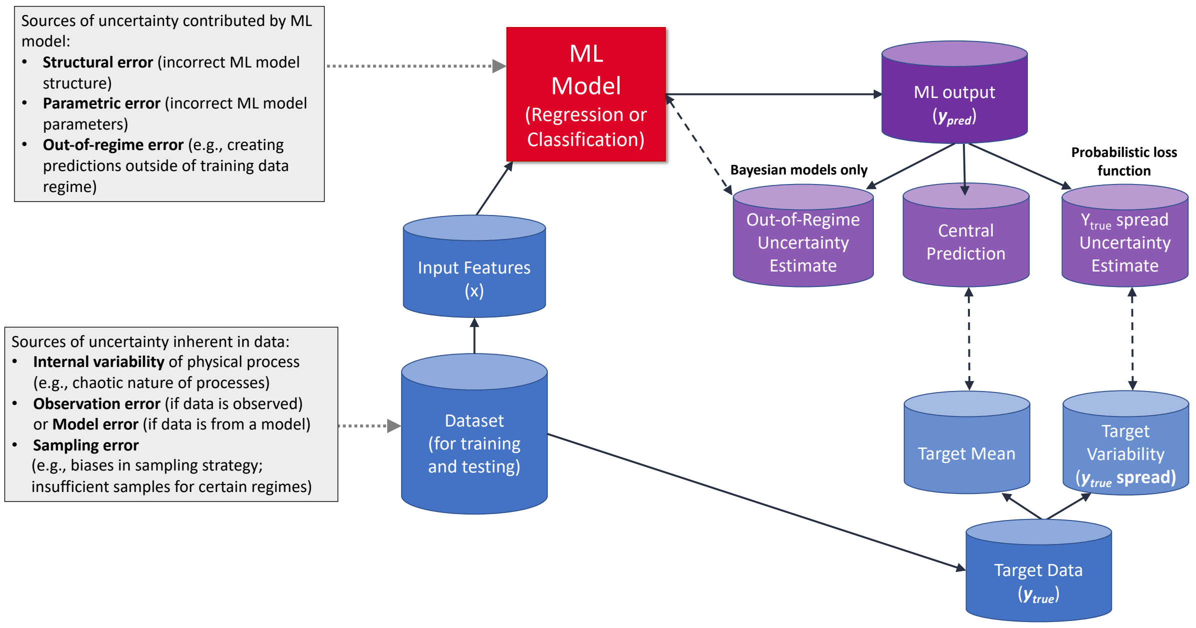
Finally, in Section 7 we provide insights on these UQ methods and metrics.

This article is accompanied by several Jupyter notebooks for implementing the methods discussed herein – both the UQ methods themselves and methods to evaluate the resulting uncertainty estimates (see `https://github.com/thunderhoser/cira\_uq4ml`). For classification tasks, a Monte Carlo Notebook (`mc_dropout_for_classification.ipynb`) implements the Monte Carlo dropout method and a Quantile Regression Notebook (`quantile_regression_for_classification.ipynb`) implements quantile regression with a special NN architecture that prevents quantile-crossing. Both of these also implement the spread-skill plot and discard test. For regression tasks, a CRPS Notebook (`crps_loss.ipynb`) implements the continuous ranked probability score (CRPS) as a loss function. Additionally, a Regression Notebook (`uq_regression.ipynb`) implements three of the UQ methods and demonstrates all four metrics. Rather than providing the specific data for the regression task, we created six different sample data sets that highlight the pros and cons of each method, allowing users to compare and contrast between the various methods and metrics.

2. Background

a. Which uncertainties are we trying to capture?

The motivation for quantifying uncertainty in ML is to provide information as to how much to trust the information provided by the model. To identify what uncertainties we can expect ML models to provide, first we need to take a look at the components of the model, sources of uncertainty, and pieces of information the model can use to quantify uncertainty (Fig. 1). In the ML framework, data is passed into the ML model as features. During training, the ML model learns to generate optimized predictions by minimizing differences between the predicted and target values using a loss function. At runtime, deterministic ML models create single-value predictions per input that correspond to each target value. This approach works in an idealized situation, where each input has a unique target, the model can perfectly learn to predict the targets, and the input data distribution during operational use never deviates from the training dataset. However, this is often not the case: datasets include numerous sources of uncertainty causing there to be a scattering of different output possibilities per input, different model structures and parameters may compromise



Sources of uncertainty contributed by ML model:
• Structural error (incorrect ML model structure)
• Parametric error (incorrect ML model parameters)
• Out-of-regime error (e.g., creating predictions outside of training data regime)
ML Model (Regression or Classification)
ML output (y_{pred})
Bayesian models only
Probabilistic loss function
Out-of-Regime Uncertainty Estimate
Central Prediction
Y_{true} spread Uncertainty Estimate
Input Features (x)
Sources of uncertainty inherent in data:
• Internal variability of physical process (e.g., chaotic nature of processes)
• Observation error (if data is observed) or Model error (if data is from a model)
• Sampling error (e.g., biases in sampling strategy; insufficient samples for certain regimes)
Dataset (for training and testing)
Target Mean
Target Variability (y_{true} spread)
Target Data (y_{true})

Figure 1: To indicate how much users should trust ML model predictions, UQ approaches seek to estimate the total uncertainty present in each model output. Currently, UQ methods provide estimates for the y_{true} spread and out-of-regime uncertainty.

model performance, and operational use may ask for predictions on input values that do not exist in the training dataset.

While ideally model uncertainty estimates would include all the discrepancy contributions, the only information with which to derive these estimates considered here are the data and the ML model. Thus, we task the ML method with quantifying two sources of uncertainty as shown in Fig. 2. The first is an uncertainty estimate capturing the data uncertainty that results from the internal variability of physical processes, which manifests as target discrepancies (Fig. 2 Error 1; Fig. 1 far right purple container). The model can learn the spread in the target data and quantify this uncertainty per individual prediction. The second is to use the model itself to learn the training data limits and to provide higher uncertainty estimates when the model is being tasked to create predictions that are outside of the training data regime, which we refer to as out-of-regime error (Fig. 2 Error 2; Fig. 1 far left purple container).

While we would like ML models to simultaneously make predictions and quantify the uncertainties resulting from the errors discussed above, it should be emphasized that the first step is always to seek to reduce these errors as much as possible.

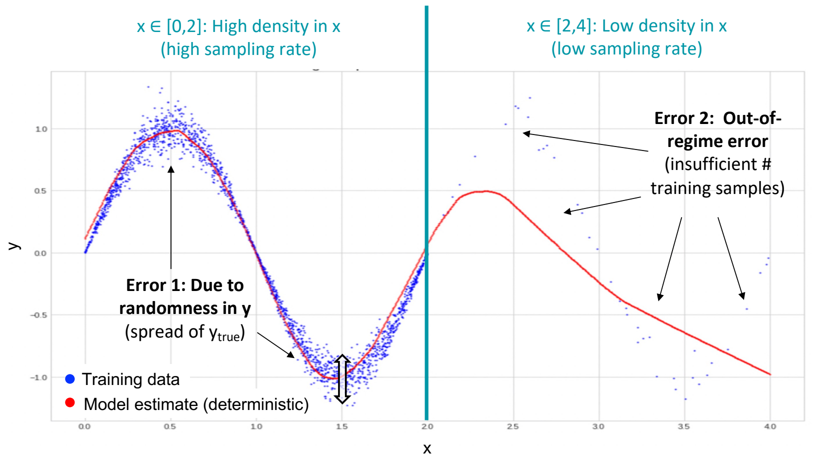


Figure 2: Illustrative example using synthetic data, where $y(x)$ is normal distributed for $x \in [0,4]$ with mean, $\mu(x) = \sin(x\pi)$, and standard deviation, $\sigma(x) = 0.1\mu(x)$. The training data has two issues: (1) varying randomness in y and (2) varying sampling rate in x. An estimate by a deterministic NN is shown in red, and the two errors we are concerned about are stated in black.

Thus, it is essential to pre-process the data and to take care when creating the dataset to minimize uncertainty sources. Similarly, it is essential to optimize the model architecture and hyperparameters (options chosen by the user) to achieve the best performance possible.

b. Conveying uncertainty

There are many ways to convey uncertainty for an estimated value, y_{pred}, such as:

1. Estimating an ensemble of predictions (*e.g.*, estimates for y_{pred} are $\{y_1, \ldots, y_k\}$).

2. Choosing a probability distribution type and estimating its parameters (*e.g.*, $\mathcal{N}(\mu,\sigma)$ with estimated values for the mean, μ, and the standard deviation, σ).

3. Estimating specific points of a probability distribution (*i.e.*, quantiles).

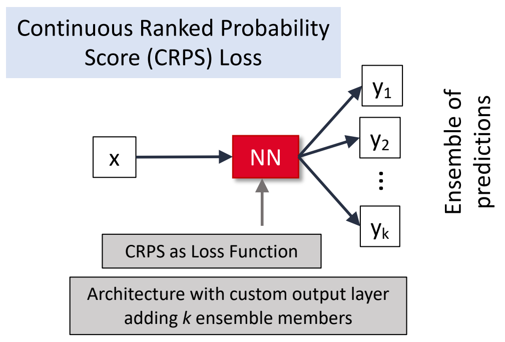
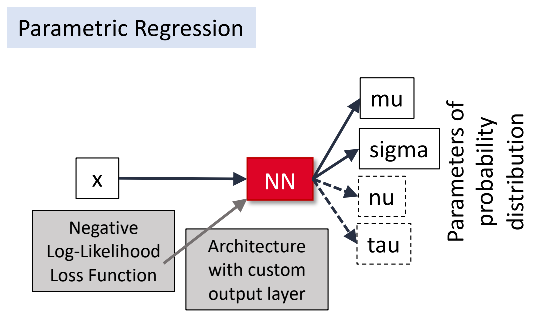
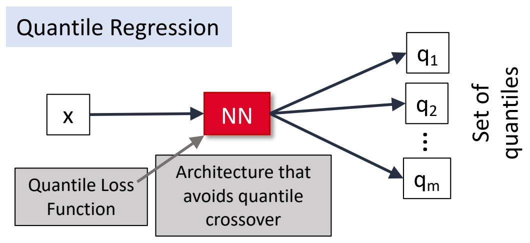
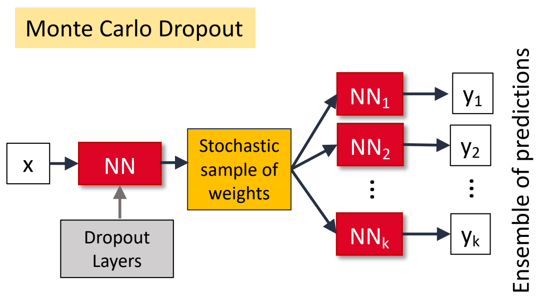
UQ Method Schematic	Key Ideas
 Continuous Ranked Probability Score (CRPS) Loss x NN y_1 y_2 y_k CRPS as Loss Function Architecture with custom output layer adding *k* ensemble members Ensemble of predictions	• Train NN to predict an ensemble representing data distribution - CRPS as loss function to minimize cumulative distribution (CDF) - Architecture has custom output layer incorporating *k* members • **Output:** Ensemble of predictions • **Tasks:** Regression • **Pros:** Fully represents the target distribution, useful for tasks with different co-occurring target regimes because it predicts the likelihood of the different outcomes, requires minimal model changes, predicts y_{true} spread • **Cons:** Number of ensemble members must be specified *a priori*, requires model architecture adjustment and custom loss • **Example:** CRPS Notebook, Regression Notebook
 Parametric Regression x NN mu sigma nu tau Negative Log-Likelihood Loss Function Architecture with custom output layer Parameters of probability distribution	• Train NN to predict parameters of a probability distribution - Type of probability distribution selected *a priori* - Negative log-likelihood loss function - Architecture with custom output layer • **Output:** Parameters of the specified probability distribution • **Tasks:** Regression • **Pros:** Requires minimal model changes, predicts y_{true} spread • **Cons:** Probability distribution must be specified *a priori*, requires model architecture adjustment and custom loss function, does not capture out-of-regime uncertainty • **Example:** Barnes et al. (2021), Regression Notebook
 Quantile Regression x NN q_1 q_2 q_m Quantile Loss Function Architecture that avoids quantile crossover Set of quantiles	• Train NN to predict a set of quantiles - Quantile loss function - Architecture that avoids quantile cross over • **Output:** Set of quantiles • **Tasks:** Classification and regression • **Pros:** Requires minimal model changes, predicts y_{true} spread • **Cons:** Number of quantiles must be specified *a priori*, requires model architecture adjustment and custom loss function, does not capture out-of-regime uncertainty • **Example:** Quantile Regression Notebook
 Monte Carlo Dropout x NN Stochastic sample of weights NN_1 NN_2 NN_k y_1 y_2 y_k Dropout Layers Ensemble of predictions	• NN weights are probability distributions, not single numbers • For every prediction call, the model randomly selects weights - Each inference is different (i.e., a new ensemble drawn) • **Output:** Ensemble of predictions - Create *k* ensemble members by running the model *k* times • **Tasks:** Classification and regression • **Pros:** Requires no model changes (except the inclusion of dropout layers), predicts out-of-regime uncertainty • **Cons:** Monte Carlo required at runtime, does not capture y_{true} spread with traditional loss function • **Example:** Monte Carlo Notebook, Regression Notebook

Figure 3: Summary schematics and key ideas for the uncertainty quantification methods. Bayesian techniques are yellow and non-Bayesian are blue.

All of these descriptions can be converted to an estimated probability distribution, which provides an estimate of uncertainty. In the remainder of this article we refer to this estimated probability distribution as *predicted distribution*.

If a scalar measure of uncertainty is desired, *e.g.* for visualization, we can calculate the standard deviation, σ, from the predicted distribution, which provides an intuitive uncertainty measure for applications where the uncertainty is nearly normal distributed. Another useful quantity for visualization is to display the 95% confidence bounds, which can also reveal where the uncertainty is not normally distributed. For non-normal distributions, additional parameters such as skewness might be useful to convey the characteristics of the estimated uncertainty in more detail.

3. Methods for Uncertainty Quantification (UQ) in Neural Networks

We briefly introduce four different methods to estimate NN uncertainty, namely probabilistic prediction using the Continuous Ranked Probability Score (CRPS) as the loss function, Parametric Regression, Quantile Regression, and Monte Carlo Dropout (and by extension, Bayesian Neural Networks). All four methods are illustrated in the examples in Sections 5 and 6. While many more methods exist, we selected these methods to provide an introduction to some of the simpler UQ methods that are currently being used for environmental-science applications. Summary schematics and key ideas for each of the UQ methods are provided in Fig. 3.

a. Probabilistic Prediction Using Continuous Ranked Probability Score: Training a NN to predict a representative ensemble of predictions

The continuous ranked probability score (CRPS) is commonly used to evaluate probabilistic forecasts, comparing an ensemble forecast to either observations (which are inherently deterministic) or a deterministic "best guess" forecast by comparing their cumulative probability distributions (CDFs) (*e.g.*, Matheson and Winkler (1976), Hersbach (2000), Gneiting et al. (2005)).

The CRPS is essentially a generalization of the mean absolute error (MAE) for ensemble predictions,

$$CRPS(F, y_{true}) = \int_{-\infty}^{\infty} \left(F(y_{pred}) - \mathcal{H}(y_{pred} - y_{true})\right)^2 dy, \qquad (1)$$

where y_{pred} is the predicted value, y_{true} is the observed value, F is the predicted CDF, and $\mathcal{H}$ is the Heaviside step function,

$$\mathcal{H} := 1, \quad \text{if } (y_{pred} - y_{true}) \geq 0, \tag{2a}$$

$$:= 0, \quad \text{if } (y_{pred} - y_{true}) < 0. \tag{2b}$$

$\mathcal{H}$ is 1 if the error is positive and 0 if the error is negative. Thus, Eq. 1 is essentially the error between the predicted CDF, $F(y_{pred})$, and the observed CDF, $\mathcal{H}(y_{pred} - y_{true})$. An example of the CRPS score for both a deterministic and a probabilistic model prediction is shown in Fig. 4.

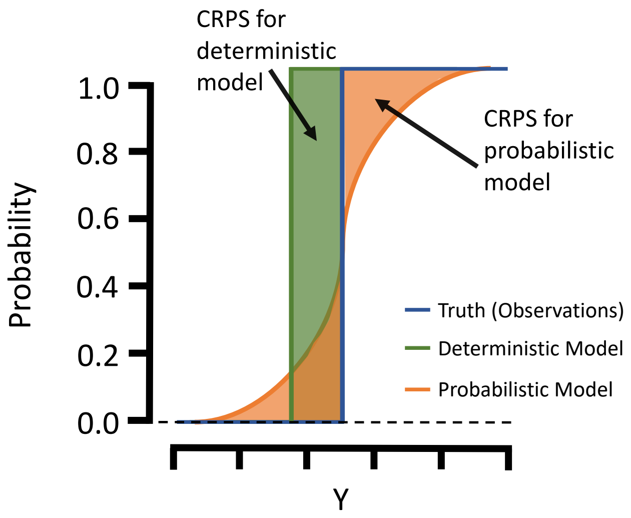


Figure 4: An example of the CRPS score for a deterministic model (green) and a probabilistic model (orange), where the truth is shown in blue. Adapted from Brey (2021).

For ML models, the CRPS plays a double role: (a) it can be used as a metric to *evaluate* UQ estimates and (b) it can be used to *generate* UQ estimates by using CRPS as a loss function to train a NN to produce an ensemble of predictions. To use the CRPS as a loss function, Gneiting and Raftery (2007) made use of distribution theory and identities from Székely and Rizzo (2005) to derive an analytical implementation of the CRPS, where

$$CRPS^{*}(F, y_{true}) = E_F\left|\hat{Y} - y_{true}\right| - \frac{1}{2} \times E_F\left|\hat{Y} - \hat{Y}'\right|. \tag{3}$$

In this equation, $CRPS^*$ is the negative orientation of $CRPS$ ($CRPS^* = -CRPS$). $\hat{Y}$ is a randomly-drawn sample of the distribution of y_{pred}, which corresponds to the predictions from all of the ensemble members, and $\hat{Y}'$ is a transposed copy of the predictions. E_F is an evaluation function in order to reduce the dimensionality to a single value, and in practice implementations often use the mean.

For normal predictive distributions, Eq. 3 can be solved directly using the mean and standard deviation; however, this can be evaluated for any distribution using Monte Carlo techniques to generate ensemble members representative of the distribution. The first term on the right side corresponds to the MAE between the NN predictions and the actual y_{true}. The second term corresponds to one half of the predicted spread, or the mean absolute value of the pairwise differences between the different ensemble members. Note that for a single ensemble member, this equation reduces to the MAE.

The negative orientation of CRPS enables the use of $CRPS^*$ as a loss function, since minimizing this function corresponds to the optimal result. A NN can then be trained to produce an ensemble of outputs, and the loss function directs the model to find an optimal distribution of the members during training. This approach removes the requirement of choosing a probability distribution *a priori*. This non-parametric formulation can be useful for meteorological applications such as predicting precipitation, which is often characterized by a mix of distributions (*i.e.*, a combination of a point-mass distribution at zero for non-events, and a gamma distribution for precipitation events).

The CRPS's utility as a loss function has been demonstrated in several environmental-science applications (*e.g.*, Rasp and Lerch (2018), Brey (2021), Grönquist et al. (2021), Scher and Messori (2021)).

An example of results from using the CRPS loss function is shown in Fig. 5. To demonstrate the utility of ensemble predictions, we used a combined exponential and point mass distribution, where the y-value for some of the data samples increases exponentially with x, while for other data samples it remains at 0. A model trained with the traditional MAE loss performs poorly for the higher x values, splitting the difference between the two data regimes and not matching any of the actual y values. This is reflected in Fig. 5B, where the CRPS score is high. In contrast, a model trained with the CRPS loss function performs much better, and the individual ensemble members are able to predict both regimes of the data, capturing both the exponential increase associated with higher x values, as well as the portion of data that remains constant at 0. This is reflected in Fig. 5B, where the CRPS score of the net CDF from the CRPS model is much lower. The mean of the ensemble predictions is also shown in Fig. 5A (heavy orange line), and similar to the MAE model, it does not match any of the data. This is important to keep in mind when evaluating ensemble predictions and dealing with dispersive data, namely that larger-scale statistical metrics are not always appropriate or the best tools to use when evaluating model performance.

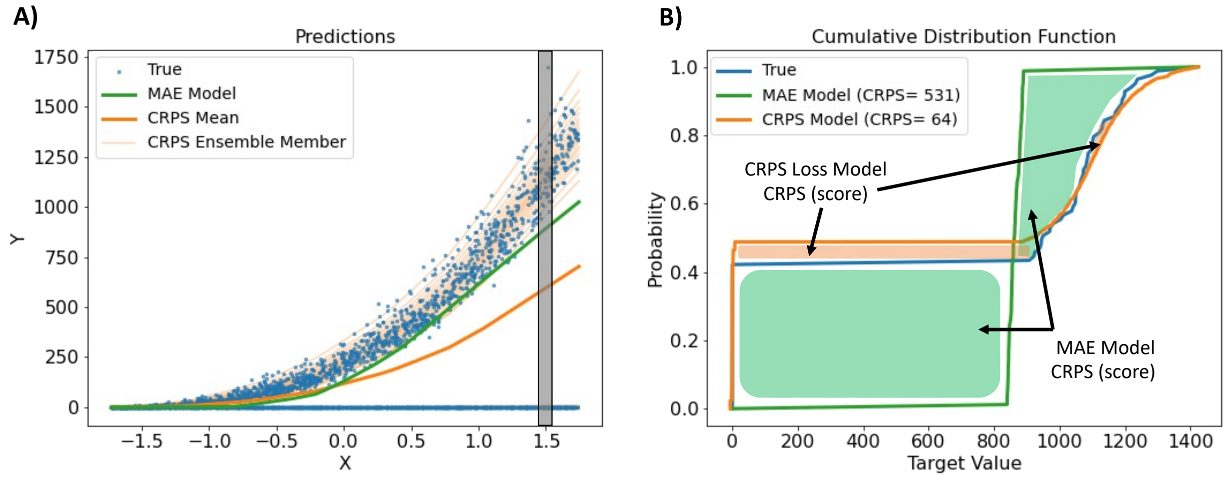


Figure 5: An example of the CRPS score for a data set drawn from exponential and point mass distributions and modeled by two NN models. The first model is deterministic and trained with the MAE as the loss. The second model is an ensemble model utilizing the CRPS as the loss function. A) Sample test data (blue dots), corresponding MAE model predictions (green line), and the corresponding CRPS model predictions for all ensemble members (light orange lines). Note that numerous ensemble members are hidden beneath the point mass distribution for the fraction of the data set where $y = 0$. The CRPS ensemble mean is also shown (heavy orange). B) The corresponding CDFs sampled at $x = 1.5$ for the data (blue line), MAE model (green line), and CRPS model (orange lines). The CRPS score for the MAE model and for the net CDF from the CRPS model ensemble members is provided in the legend. Note that in the loss function implementation, the score is the mean of the CRPS scores for each ensemble member, rather than the score for the CDF resulting from combining the ensemble members. While this results in higher (unfavorable) scores during training, this implementation allows for better optimization of each ensemble member.

b. Parametric Regression: Training a NN to predict parameters for a predefined type of probability distribution

A simple approach for adding uncertainty to regression predictions is to train a neural network to predict a probability distribution of user-specified type (*e.g.*, normal distribution). This results in a probabilistic estimate for each input sample instead of a single number. Barnes et al. (2021) implemented this method, using neural networks to predict both a normal distribution and a sinh-arcsinh (SHASH) distribution. When choosing the normal distribution to represent local uncertainty, the model predicts both the mean value as well as the standard deviation, with the latter representing the uncertainty for the specific prediction (*e.g.*, Fukushima and Miyake (2022)).

The SHASH distribution is a more general distribution that allows for representation of asymmetrical and/or heteroscedastic data, and the model predicts four parameters to determine the distribution's location, scale, skewness, and tailweight. Python code demonstrating this technique is provided in the Regression Notebook.

Rather than explicitly predicting the parameters, another way to implement this approach is to directly predict a distribution, which is now supported with the TensorFlow Probability library (Dillon et al. 2017). In addition to providing the parameters, TensorFlow distributions include methods to create samples and calculate statistical metrics. When doing this, the loss function is the negative log-likelihood of the distribution (Salama 2021).

To demonstrate this approach, Fig. 6 demonstrates NNs predicting the normal and SHASH distributions on a sample dataset. From Fig. 6B and C, both NNs are able to predict the mean, larger uncertainties where the data spread is larger (i.e., $x < 2$ and $6 < x < 8$, and smaller uncertainties when the data spread is small (i.e., $x \approx 3$). Zooming in on predictions centered on $x = 1$, Fig. 6D shows the true data spread and the predicted distributions. Since the spread is normally distributed for this value of x, both models capture the true distribution well and create nearly identical predictions. This is also the case looking at $x = 3$ (Fig. 6E), where both models accurately predict smaller uncertainty spread.

Differences between the two models appear when looking at the predicted uncertainty ranges for data spread that is not normally distributed. To demonstrate this, focus on the region $6 < x < 8$, where the data spread is skewed. For the Normal NN, the confidence intervals are symmetrical about the mean, causing the confidence bounds to mismatch the true data spread (Figs. 6B and F). In contrast, because the SHASH NN has more degrees of freedom, it is able to more closely match the data spread by predicting a skewed distribution (Fig. 6C and F). This example shows that predicting a SHASH distribution can be advantageous in places where the data spread does not follow a normal distribution; however, in practice, we found that due to data limitations and small learning rates, the normal distribution should be used instead of SHASH unless the distribution is known to be non-normal *a priori*.

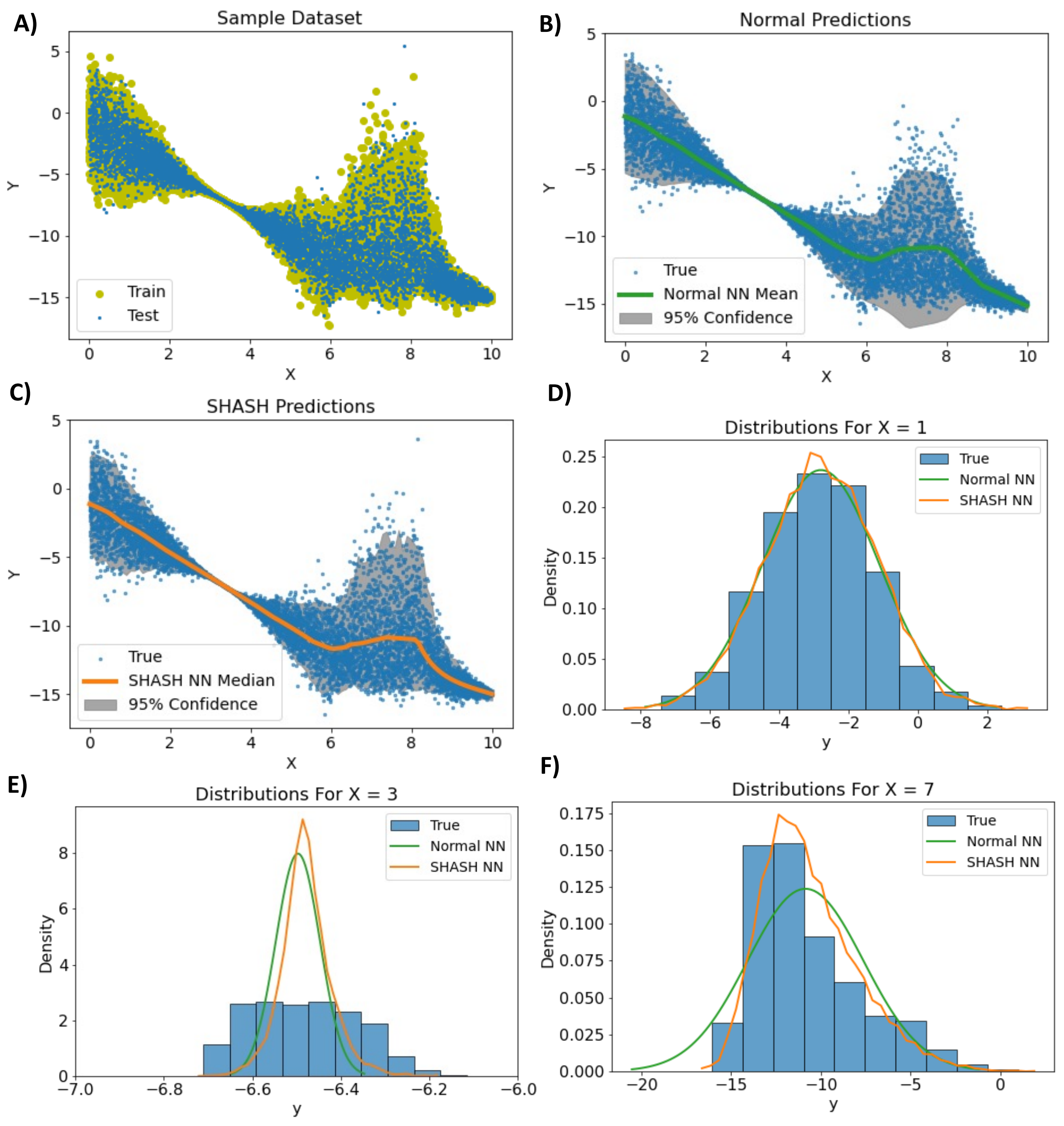


Figure 6: An example of uncertainty estimates from models trained to predict normal distribution parameters (Normal NN) and SHASH distribution parameters (SHASH NN). A) Sample dataset. B) The Normal NN model mean predictions and 95% confidence bounds. C) The SHASH NN model median predictions and 95% confidence bounds. D) Sample distributions for $x = 1$. The histogram of the test values is shown in the blue bars, the predicted normal distribution by the Normal NN is shown in the green line, and the predicted distribution by the SHASH NN is shown in the orange line. E) Same as D, but for $x = 3$. F) Same as D, but for $x = 7$.

c. Quantile regression: Directly predicting quantiles

Quantile regression (QR) involves directly predicting the quantiles of a probability distribution. This means that for each data sample, instead of predicting a single number (the mean or "expected value" or "maximum-likelihood estimate"), we predict several numbers (quantile-based estimates).

Recently, QR has gained wide popularity for NNs (*e.g.*, Taylor 2000; Cannon 2011; Yu et al. 2020). The "trick" is to train the NN with the quantile loss function:

$$\mathcal{L} = \begin{cases} q \quad |y_{\text{true}} - y^q_{\text{pred}}| \quad , & \text{if } y_{\text{true}} > y^q_{\text{pred}}; \\ (1-q)|y_{\text{true}} - y^q_{\text{pred}}| \quad , & \text{if } y_{\text{true}} \leq y^q_{\text{pred}}. \end{cases} \tag{4}$$

Here, q is the desired quantile level, ranging from $[0,1]$; y_{true} is the correct value; and y^q_{pred} is the estimated value at quantile level q. Large values of q penalize underprediction ($y^q_{\text{pred}} < y_{\text{true}}$) more than overprediction ($y^q_{\text{pred}} > y_{\text{true}}$), encouraging the model to output large y^q_{pred}. Conversely, small values of q encourage the model to output small y^q_{pred}.

To estimate multiple quantiles with NNs, a common approach is to train a separate NN for each quantile. Because the different NNs are trained independently, this approach does not prevent the problem of quantile-crossing, where the estimated value y^q_{pred} decreases as the quantile level q increases (*e.g.*, the 25th-percentile rainfall prediction is 30 mm but the 75th-percentile prediction is 20 mm). Thus, we have developed a novel NN architecture that prevents quantile-crossing. For any consecutive pair of quantile levels, q_{i-1} and q_i, the estimate $y^{q_i}_{\text{pred}}$ must be $\geq y^{q_{i-1}}_{\text{pred}}$. To satisfy this condition, we express $y^{q_i}_{\text{pred}}$ as the sum of the previous quantile-based estimate, $y^{q_{i-1}}_{\text{pred}}$, and a positive term. Specifically, we implement the following equation:

$$y^{q_i}_{\text{pred}} = y^{q_{i-1}}_{\text{pred}} + \text{ReLU}(\Delta y^{q_i}_{\text{pred}}), \tag{5}$$

where ReLU is the rectified linear unit (Nair and Hinton 2010), an activation function commonly used for NNs, defined as $\text{ReLU}(w) = \max(0, w)$.

Instead of estimating $y^{q_i}_{\text{pred}}$ directly, we train the NN to estimate $\Delta y^{q_i}_{\text{pred}}$, then implement Eq. 5 inside the NN architecture to obtain $y^{q_i}_{\text{pred}}$. For Python code, see the Quantile Regression Notebook.

For a schematic representation, see Fig. 16b and the accompanying discussion in Section 6a.

d. Monte Carlo Dropout and Bayesian Neural Networks: Stochastic NNs Utilizing Ensemble Learning

Dropout was invented as a regularization method (Hinton et al. 2012) to prevent overfitting in NNs. Dropout is applied to one NN layer at a time (though it may be applied to multiple layers in the same NN). During each NN forward pass, letting the dropout rate be d and number of neurons in the layer be N, Nd neurons are randomly dropped out, leaving the remaining $N(1-d)$ neurons to represent useful features of the input data. The specific neurons dropped out are different for each forward pass. Thus, any subset of $N(1-d)$ neurons must be able to represent input features adequately, *i.e.*, in a way that yields a skillful prediction. This forces the neurons to learn more independently of each other, creating a pseudo-ensemble. In common practice, dropout is used only during training; at inference time all neurons are used, making the NN deterministic. However, dropout can also be used at inference time, making the NN stochastic and producing a predictive *distribution* by running the NN many times, which is called Monte Carlo (MC) dropout (Gal and Ghahramani 2016).

The main advantage of MC dropout is ease of implementation – Seoh (2020) described it as "shockingly simple". Dropout is a pre-defined layer in Keras, and passing the argument `training=True` during model construction ensures that dropout will be used at inference time, as shown in the Monte Carlo and Regression Notebooks. Disadvantages of MC dropout are that brute-force sampling (*i.e.*, running the NN many times in inference mode) is computationally expensive and the correct hyperparameters (which dropout rates to use and in which layers) are unclear. Furthermore, although it is a good way to estimate out-of-regime uncertainty, MC dropout alone does not capture the full spread in y_{true} (Bihlo 2021; Klotz et al. 2021; Garg et al. 2022). However, MC dropout can easily be combined with other UQ methods to provide uncertainty estimates for both data spread and out-of-regime errors (*e.g.*, Sato et al. (2021) and Yagli et al. (2022)).

Bayesian Neural Networks (BNNs) are quite complex - both conceptually and computationally - and beyond the scope of this introductory article. Jospin et al. (2022) provide an excellent tutorial for BNNs, and Orescanin et al. (2021) and Ortiz et al. (2022) demonstrate their use for remote sensing tasks.

Despite their complexity, BNNs may provide a more robust estimate of the out-of-regime uncertainty and are more flexible than MC dropout (Jospin et al. 2022; Salama 2021). While Bayesian methods, such as MC dropout and BNNs, are necessary to capture out-of-regime error estimates, how well they do in practice remains on open research question.

4. Methods for evaluating uncertainty estimates

From an estimated probability distribution, one can derive the central prediction (or "best guess"; usually the mean) and uncertainty (or "spread"; often the standard deviation or 95% confidence interval). Although our main focus is on evaluating uncertainty estimates, one should not forget to evaluate the central prediction. This section discusses four evaluation methods, among which one (the attributes diagram) pertains to the central prediction and the other three pertain to uncertainty. This section concludes with Table 1, which summarizes the information each evaluation method conveys and provides situations where it may be used. While single-number summaries like the CRPS discussed in Section 3a make useful loss functions, they are less useful for evaluating a trained model, where richer information is desired.

a. The attributes diagram for binary classification

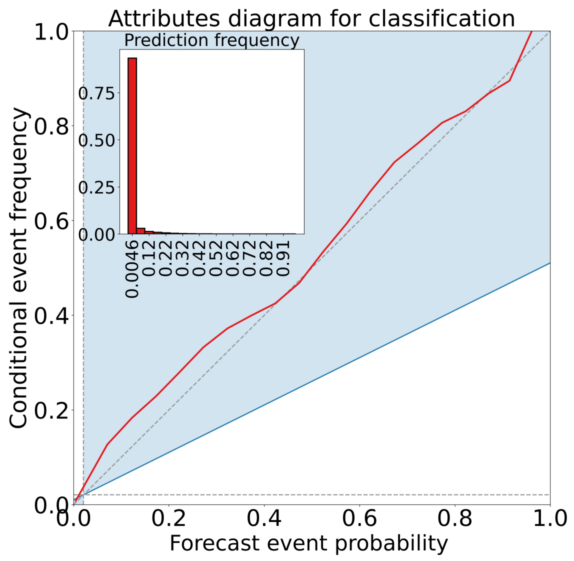


Figure 7: Attributes diagram for binary classification. The inset histogram shows how often each probability is predicted. The {diagonal, horizontal, vertical} grey dashed line is the {perfect-reliability, no-resolution, climatology} line; the polygon shaded in blue is the positive-skill area; and the red line is the reliability curve.

The attributes diagram (Fig. 7) is a reliability curve with additional information. The reliability curve plots model-predicted event probability[1] (henceforth, just "event probability") on the x-axis versus conditional observed event frequency (henceforth, just "conditional event frequency") on the y-axis. Each point corresponds to one bin of event probabilities. For example, suppose that the event is tornado occurrence; the reliability curve uses 10 probability bins, equally spaced from 0.0 to 1.0; and one point on the curve is $(x = 0.15, y = 0.4)$. This means that in cases where the model predicts a tornado probability between 10% and 20%, a tornado actually occurs 40% of the time. The reliability curve is used to identify conditional model bias, *i.e.*, bias as a function of the event probability. Points below (above) the 1-to-1 line show probabilities at which the model overpredicts (underpredicts). For a perfectly reliable model, the reliability curve follows the 1-to-1 line. In other words, when the event probability is p%, the actual event occurs p% of the time[2].

The full attributes diagram (Hsu and Murphy 1986) contains the reliability curve plus four additional components. The first is the perfect-reliability line, or 1-to-1 line. The second is the no-resolution line, a horizontal line at $y = \overline{y}_{\text{climo}}$, where $\overline{y}_{\text{climo}}$ is the event frequency computed over the full dataset, called "climatology". If the reliability curve follows the no-resolution line, the conditional event frequency is always $\overline{y}_{\text{climo}}$, regardless of the event probability; thus, the model is completely uninformative on the expected observation. The third is the climatology line, a vertical line $x = \overline{y}_{\text{climo}}$. For the climatological model (which always predicts probability = $\overline{y}_{\text{climo}}$), the reliability curve is a single point at $x = y = \overline{y}_{\text{climo}}$, or the intersection of the climatology and no-resolution lines. The fourth is the positive-skill area, a polygon defining where the Brier skill score (BSS) is positive. The BSS is defined as $\frac{\text{BS}_{\text{climo}} - \text{BS}}{\text{BS}_{\text{climo}}}$, where BS and BS_{climo} are the Brier scores of the model of interest and the climatological model. The BSS ranges from $(-\infty, 1]$, and values > 0 signal an improvement over climatology.

b. The attributes diagram for regression

Although the attributes diagram is typically used for binary classification, it can also be adapted for regression (Fig. 8). Letting the target variable be z, the x-axis is the predicted z-value and the y-axis is the conditional mean observed z-value, both real numbers that in general can range from

[1]When ML models are used for classification, by default they produce confidence scores ranging from $[0, 1]$, which are not true probabilities. However, in this paper we adopt common parlance and refer to confidence scores as probabilities.

[2]The reliability curve uses only the central prediction – *i.e.*, the x-coordinate is the mean of the predictive distribution, not a measure of uncertainty. However, UQ studies commonly employ the reliability curve to evaluate the central prediction (*e.g.*, Delle Monache et al. 2013; Jospin et al. 2022; Chapman et al. 2022), so we include it in this paper.

$(-\infty, +\infty)$. The perfect-reliability line is still the 1-to-1 line; the no-resolution line is at $y = \overline{z}_{\text{climo}}$; and the climatology line is at $x = \overline{z}_{\text{climo}}$. The interpretation of the perfect-reliability, climatology, and no-resolution lines is the same. The positive-skill area shows where the mean squared error (MSE) skill score (MSESS) is positive.

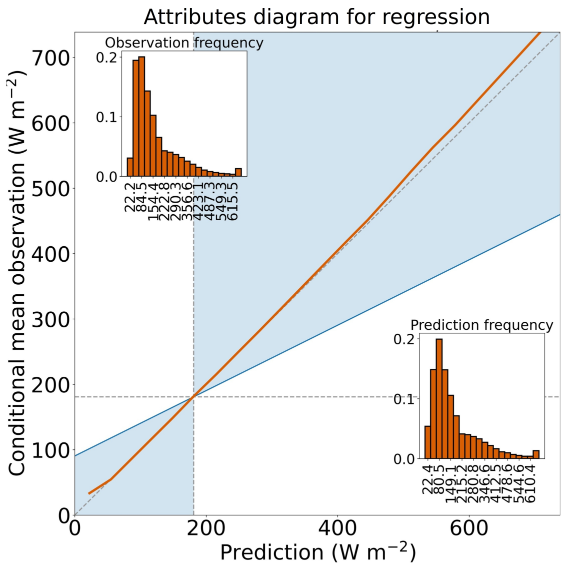


Figure 8: Attributes diagram for regression – in this particular case, for predicting radiative flux in W m^{-2}. The inset histogram shows how often each flux is predicted. The {diagonal, horizontal, vertical} grey line is the {perfect-reliability, no-resolution, climatology} line; the polygon shaded in blue is the positive-skill area; and the red line is the reliability curve.

c. The spread-skill plot

The spread-skill plot (for which we owe much of our understanding to Delle Monache et al. 2013) evaluates uncertainty estimates, thus it may be used *only* for models that include uncertainty. Conceptually, the spread-skill plot answers the question: "For a given predicted model spread, what is the actual model error?" The spread-skill plot shows the predicted model spread on the x-axis versus the actual model error on the y-axis (Fig. 9). Specifically, the x-axis is the mean standard deviation ($\overline{\text{SD}}$) of the model's predictive distribution, while the y-axis is the root mean squared error (RMSE) of the model's mean prediction. Each red point corresponds to one bin of spread values. For a regression problem, letting the target variable be z, the two quantities are defined as follows for the k^{th} bin:

$$\begin{cases} \text{RMSE}_k = \left[\frac{1}{N_k} \sum_{i=1}^{N_k} (z_i - \overline{\hat{z}_i})^2 \right]^{\frac{1}{2}}; \\ \overline{\text{SD}}_k = \frac{1}{N_k} \sum_{i=1}^{N_k} \left[\frac{1}{M-1} \sum_{j=1}^{M} (\overline{\hat{z}_i} - \hat{z}_{ij})^2 \right]^{\frac{1}{2}}; \\ \overline{\hat{z}_i} = \frac{1}{M} \sum_{j=1}^{M} \hat{z}_{ij}. \end{cases} \tag{6}$$

where z_i is the observed value for the i^{th} example; $\overline{\hat{z}_i}$ is the mean prediction for the i^{th} example; $\hat{z}_{ij}$ is the j^{th} prediction for the i^{th} example; N_k is the total number of examples in the k^{th} bin; and M is the total number of predictions per example[3]. The spread-skill plot can also be used for classification, replacing z with y (a binary label) and $\hat{z}$ with p (a probability).

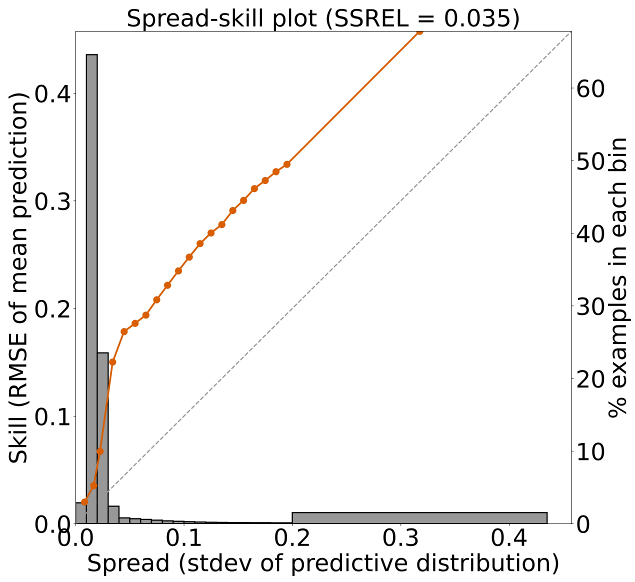


Figure 9: Spread-skill plot. The dashed line represents a perfect spread-skill plot, and the histogram shows how often each spread value occurs.

For a model with perfectly calibrated uncertainty estimates, the spread-skill plot follows the 1-to-1 line. At points below the 1-to-1 line ($\overline{\text{SD}}$ values where $\overline{\text{SD}} >$ RMSE), the model is overspread or "underconfident"; at points above the 1-to-1 line ($\overline{\text{SD}}$ values where $\overline{\text{SD}} <$ RMSE), the model is underspread or "overconfident". The overall quality of the spread-skill plot can be summarized by

[3]Eqs. 6 assume that the model explicitly provides an ensemble of M predictions. Some models quantify uncertainty in other ways, *e.g.*, by specifying the quantiles or parameters of a probability distribution. However, in the latter two cases – and in general – there is still a way to compute the mean and standard deviation of the predictive distribution, regardless of how the model quantifies uncertainty.

mean distance from the 1-to-1 line, which we call the spread-skill reliability (SSREL). SSREL is defined as:

$$\text{SSREL} = \sum_{k=1}^{K} \frac{N_k}{N} |\text{RMSE}_k - \overline{\text{SD}_k}|, \tag{7}$$

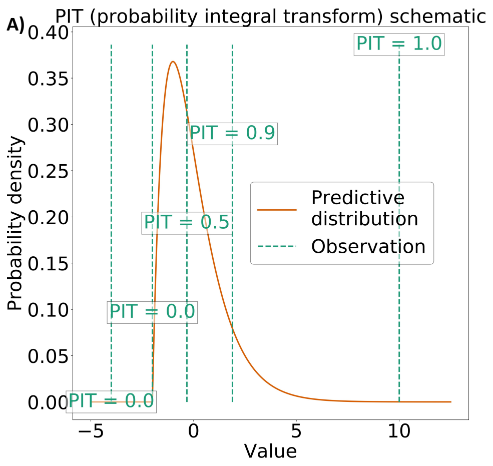
where N is the total number of examples; K is the total number of bins; and all other variables are as in Eq. 6.

d. The probability integral transform (PIT) histogram

The PIT is the cumulative distribution function (CDF) of the predictive distribution, evaluated at the observed value. Mathematically, letting the target variable be z, the observed value be z_{obs}, a single prediction be $\hat{z}$, and the CDF be $\text{CDF}(z)$, the PIT is defined as:

$$\text{PIT} = \text{CDF}(z_{\text{obs}}) = \text{probability}(\hat{z} \leq z_{\text{obs}}). \tag{8}$$

This can also be interpreted as the quantile of the predictive distribution where the observed value occurs. A few examples are shown in Fig. 10a. Note that the PIT is meaningful only for regression problems, not for binary classification. For binary classification the only possible observations are 0 and 1, while predictions (event probabilities) must range from $[0, 1]$. Thus, z_{obs} always occurs at one extreme of the predictive distribution, so the only possible PIT values are 0 and 1. Intermediate PIT values do not occur, which makes for a trivial PIT histogram.



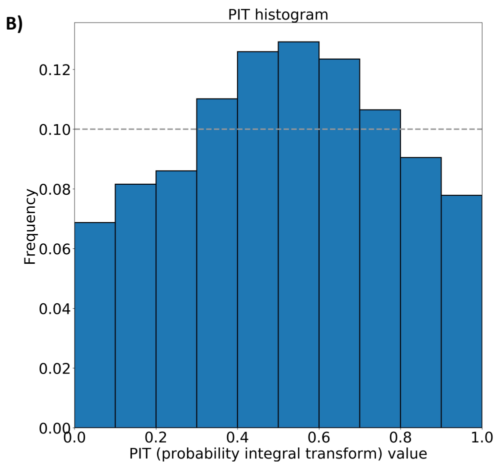


Figure 10: [a] Schematic explaining the probability integral transform (PIT). [b] Example of PIT histogram. The dashed line represents a perfect PIT histogram.

The PIT *histogram* plots the distribution of PIT over many examples, with one PIT value per example (Fig. 10b). For a perfectly calibrated model, all PIT values occur equally often, so the histogram is uniform. If the histogram has a hump in the middle (as seen in Fig. 10b), there are too many examples with intermediate PIT values (where z_{obs} occurs near the middle of the predictive distribution) and too few examples with extreme PIT values (where z_{obs} occurs near the end of the predictive distribution), so the extremes of the predictive distribution are generally too extreme. In other words, the predictive distribution is generally too wide, so the model is overspread or "underconfident". If the histogram has humps at the ends, the predictive distribution is generally too narrow, so the model is underspread or "overconfident". The PIT histogram is a generalization of the rank histogram (or "Talagrand diagram"), which is more familiar to atmospheric scientists (Hamill 2001) and can be interpreted the same way.

e. The discard test

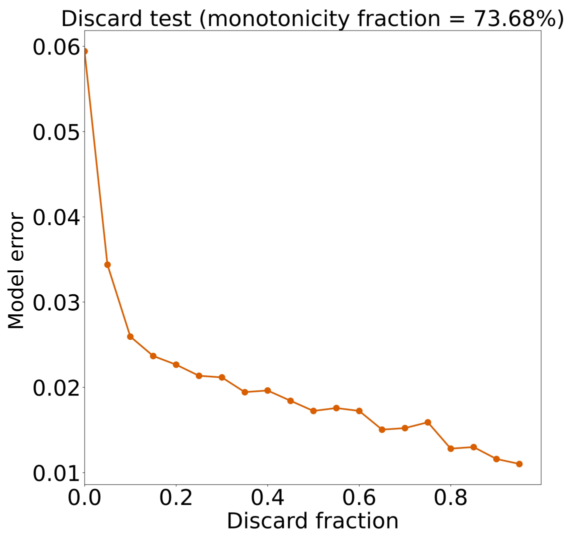


Figure 11: Sample results for discard test.

The discard test, inspired by Barnes and Barnes (2021) and similar to the filter experiment in Fig. 8.18 of Dürr et al. (2020), compares model error versus the fraction of highest-uncertainty cases discarded. For a model with well calibrated uncertainty estimates, error should decrease monotonically as the discard fraction increases. The overall quality of the discard test can be

Table 1: Methods used to evaluate the central prediction (attributes diagram) and uncertainty (all others). A check mark under "Class?" indicates whether the evaluation method can be used for classification models, and "Reg?" indicates whether it can be used for regression models.

Method	Class?	Reg?	What it tells us
Attributes diagram	✓	✓	Class: observed event frequency as a function of predicted event probability, Brier score, Brier skill score Reg: mean observed target value as a function of predicted target value, mean squared error (MSE), MSE skill score If central prediction is perfectly calibrated, this plot follows the 1-to-1 line. (Regression Notebook)
Spread-skill plot	✓	✓	Model error as a function of predicted model spread. If uncertainty is perfectly calibrated, this plot follows the 1-to-1 line. (Monte Carlo, QR, and Reg. Notebooks)
PIT histogram		✓	Distribution of PIT values. If uncertainty is perfectly calibrated, this distribution is uniform, so the PIT histogram is flat. (Reg. Notebook)
Discard test	✓	✓	Model error vs. discard fraction. For well-calibrated uncertainties, the error decreases monotonically as the discard fraction increases (*i.e.*, as more high-uncertainty samples are dropped). (Monte Carlo, QR, and Reg. Notebooks)

summarized by the monotonicity fraction (MF):

$$\text{MF} = \frac{1}{N_f - 1} \sum_{i=1}^{N_f - 1} \mathcal{I}\left(\epsilon_i \geq \epsilon_{i+1}\right), \tag{9}$$

where N_f is the number of discard fractions used; ϵ_i is the model error with the i^{th} discard fraction (*i.e.*, if this fraction is r, model error without the $100r\%$ of highest-uncertainty cases); and $\mathcal{I}()$ is the indicator function, which evaluates to 1 if the condition is true and 0 if the condition is false.

We note a crucial difference between evaluation methods: the discard test is concerned with the *ranking quality* of uncertainty estimates, whereas the spread-skill plot is concerned with the *value-estimate quality* of uncertainty estimates. In high-MF/high-SSREL cases (good discard test but poor spread-skill plot), the model's uncertainty estimate is well correlated with its error but not with the actual spread; in low-MF/low-SSREL cases, the opposite is true.

5. Illustration of UQ and evaluation for a regression task

a. Predicting dewpoint profiles for severe weather nowcasting

Vertical profiles of dewpoint are extremely useful in predicting deep convection that can lead to severe weather. These storms pose a lightning threat and may be accompanied by heavy rain, high winds, large hail, and tornados, all of which pose threats to both property and lives. Currently, forecasters rely on a relatively temporally and spatially sparse network of observations from radiosonde launches and numerical weather prediction (NWP) models. The goal of this work is to use ML techniques to combine information from NWP models with satellite data in order to improve dewpoint vertical profiles.

For this task, we use a 1D Convolutional Neural Network (CNN) configured as a fully-convolutional U-net (Ronneberger et al. 2015) to predict dewpoint at 256 vertical levels in the atmosphere. The predictors (input features) are initial profile predictions from the Rapid Refresh (RAP) model (Benjamin et al. 2016) and satellite data from the Geostationary Operational Environmental Satellite (GOES)-16 Advanced Baseline Imager (Schmit et al. 2017). The target data for this study are vertical profiles from radiosonde observations (RAOBs) collected over the central United States between January 1, 2017 and August 31, 2020. We use 75% of the data for training, 10% for validation, and 15% for testing. The model and experimental setup are described fully in Stock (2021), and the model architecture used in this study is shown in Fig. 12.

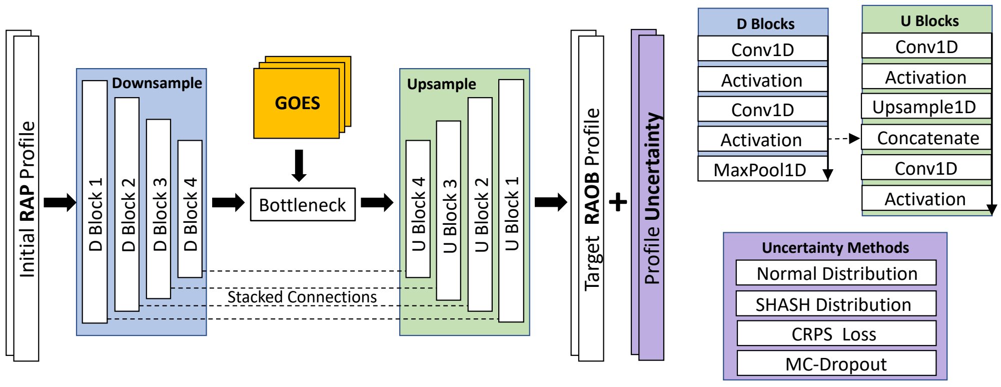


Figure 12: U-net architecture using the RAP temperature and moisture profiles as inputs and concatenating the GOES ABI (satellite) data at the bottleneck of the network. The output is the reconstructed dewpoint profile, which can be directly compared to the radiosonde observations. Additionally, uncertainty information is predicted for each layer in the profile. Adapted from Stock (2021), which did not include uncertainty estimates.

To quantify uncertainty, we train the model using four approaches. The first two are using parametric regression to predict normal and SHASH distributions (Section 3b). The third is to create probabilistic predictions using the CRPS loss function (Section 3a), and the last approach is to use MC dropout with a mean squared error (MSE) loss (Section 3d). For both the CRPS loss and MC dropout approaches, we used 60 ensemble members. For MC dropout, we used a dropout rate of 0.1 and placed dropout layers between all of the D Blocks, all of the U Blocks, and just prior to the output layer (Fig. 12). We tested dropout rates varying from 0.01 to 0.5, as well as including dropout in the D Blocks (not shown) and only saw minimal changes from the results shown.

b. Sounding profiles

Fig. 13 shows three example dewpoint profiles, as well as the mean predicted profiles. As seen in Fig. 13A-C, the uncertainty estimates from predicting normal distribution parameters appear to do a reasonable job at capturing the mismatch between the model (blue) and the observations (black). Near the surface, the predicted dewpoint has minimal uncertainty coinciding with low errors. Depending on the specific sounding, the errors increase at different levels in the profiles, and the uncertainty correspondingly increases. In addition to looking at individual profiles, it is also possible to aggregrate the results to show the mean vertical profile and uncertainty, which is shown in Fig. 13D. Here we see that on average the dewpoint profile is more certain near the surface and most uncertain in the mid- to upper-atmosphere around 500-300 mb.

c. UQ results

Here we evaluate the predictions from these methods using the different metrics in this paper (Fig. 14). Starting with the central predictions, we see in the attribution diagram (Fig. 14A) that each U-net model - regardless of the UQ method used - does a reasonable job at capturing the central dewpoint, as all of the model results overlap near the 1:1 line, with RMSEs of ~5 C. For the coldest predictions, the MC dropout has a slight high-bias, and the SHASH model may have a slight low bias around -30 C. However, overall the diagram indicates that all methods perform similarly for the dewpoint central predictions.

Moving on to evaluating the predicted uncertainty, the three different metrics in this study highlight differences between the different UQ methods. Looking at the spread-skill diagram (Fig.

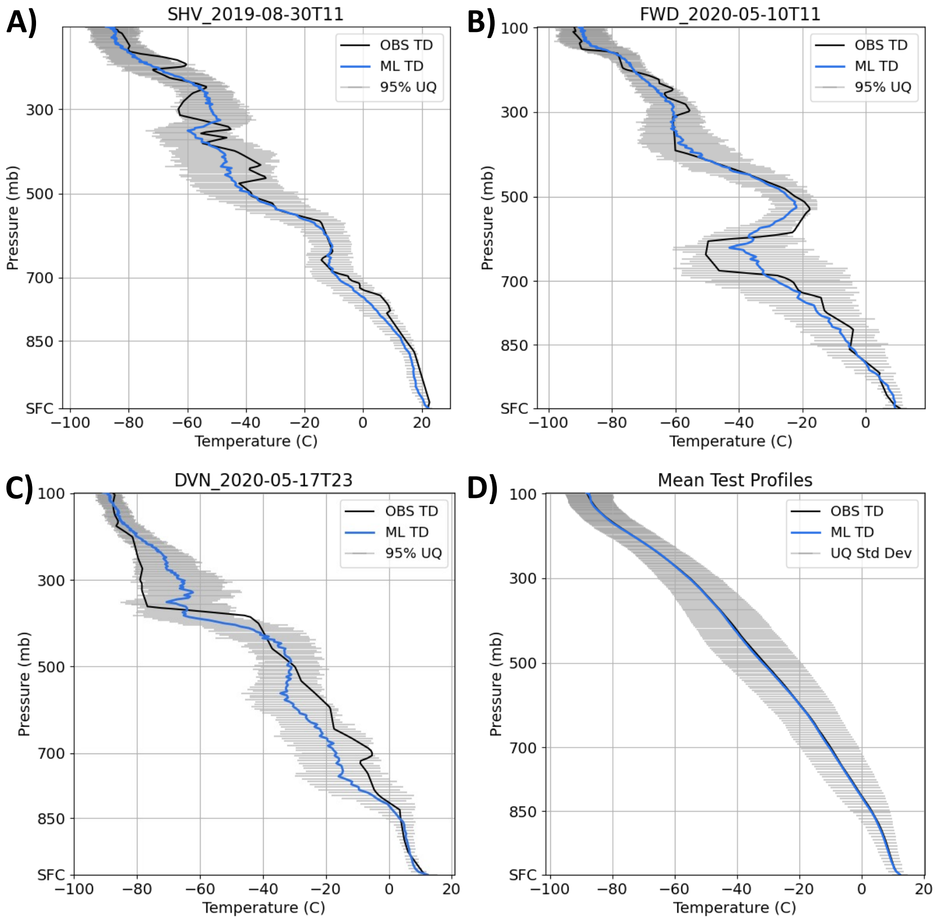


Figure 13: Sample vertical profiles of observed (black) and modeled dewpoint (TD, blue). The horizontal gray lines are the uncertainty estimates predicted using the parametric regression method. A), B), and C) Individual soundings. For these, the uncertainty shown is the 95% range of uncertainty. D) Mean vertical profiles for the testing data, where the uncertainty is the standard deviation since the profiles have been aggregated.

14B), for the vast majority of the vertical profiles, the two parametric regression methods and the CRPS model predict uncertainties matching the corresponding skill (RMSE) and fall along the 1:1 for the majority of predictions, with the spread and skill values coinciding up to predictions within 8 C. Above this, all three methods are underconfident for these relatively few soundings, and both parametric regression methods underestimate the uncertainty by ~40%. These three methods also have nearly identical frequencies as seen in the inlay, with the majority of soundings having errors $<$ 8 C. For errors larger than 8 C, these three methods overestimate the uncertainty compared to the skill, with the CRPS model being the most underconfident.

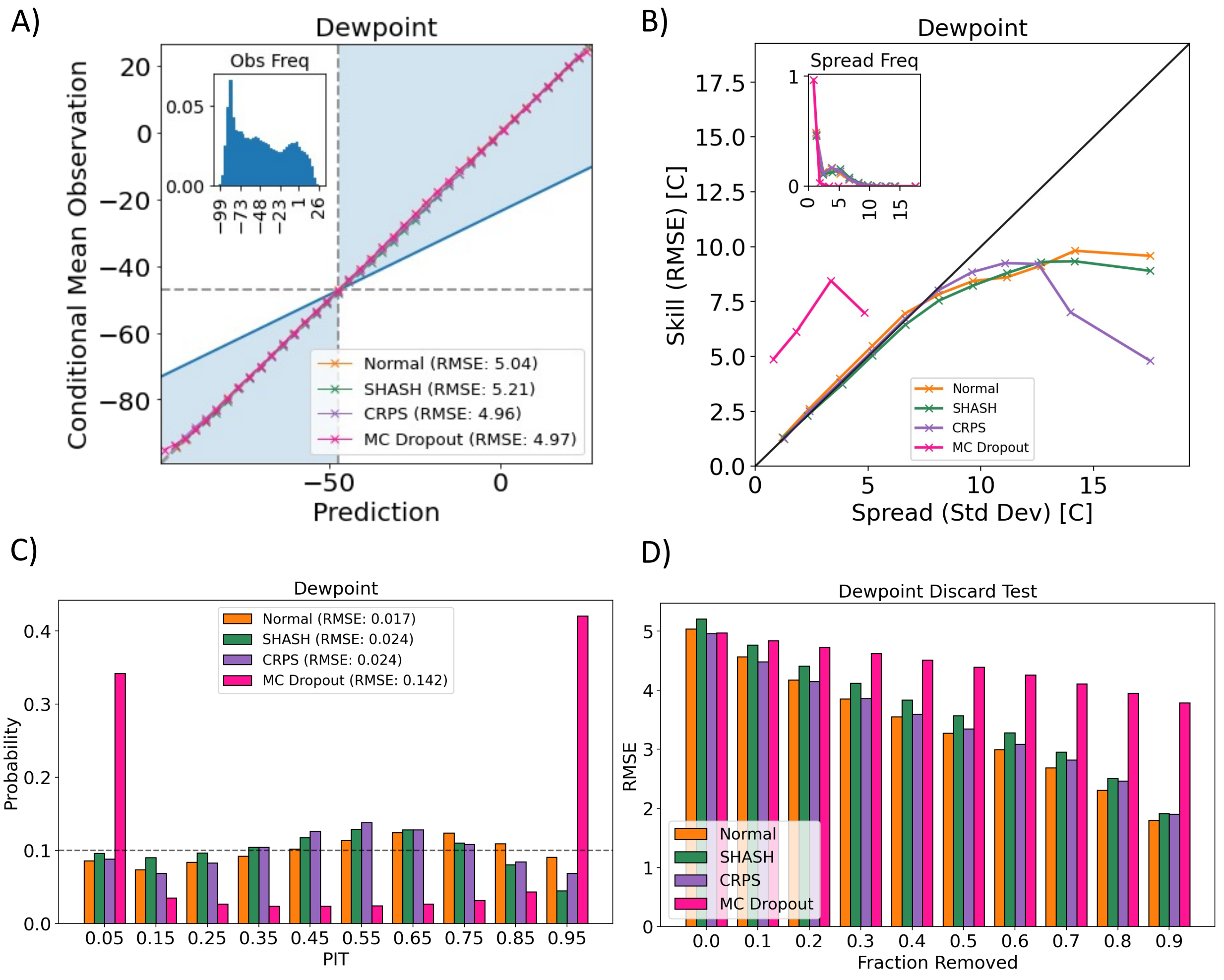


Figure 14: Dewpoint uncertainty analysis, showing results of a U-net trained with four different UQ methods, namely Normal/SHASH distribution methods from Section 3b, CRPS method from Section 3a, and MC dropout from Section 3d. A) Attribution diagram, B) Spread-Skill plot, C) PIT diagram, and D) Discard test results.

For the CRPS method, the error actually drops for the predictions with the highest spread, making the model appear to be highly underconfident. However, it is important to keep in mind that this method does not predict distribution parameters, but instead creates ensembles of predictions. For applications where all of the values fall within a given range of values, using bulk metrics such as the standard deviation to capture the spread are appropriate; however, in circumstances where it is known that a small fraction of the data largely deviates from the mean value, the evaluation becomes more complicated or even misleading. For these cases with a high spread, this diagram indicates that some of the ensemble members actually have much lower errors than the mean values from the distribution methods. This highlights an important distinction in the CRPS method that the ensemble members are not trained to just capture a distribution around the central value, but rather are trained to encapsulate all data values. While this characteristic is certainly a benefit of the method, it also adds an additional complexity to its evaluation. For tasks with rare events, it may be more appropriate to show histograms (or probabilities) of the ensemble members or to select the less-probable ensemble members and to use this, rather than including all members; however, this may be difficult to do in an operational setting. In this application, the cases with discrepant observations are rare, which is why the distribution methods have such a success and also why the central predictions of the CRPS method do not suffer nor reflect the change in mean from the deviant ensemble members (Fig. 14A).

When using the MC dropout method to quantify uncertainty, the model is overconfident. As seen in the inlay in Fig. 14B, the spread histogram is quite different from the other three methods, with the majority of predictions falling in the first bin with the lowest spread. Since most of the predictions have this low value of uncertainty, the mean errors are actually larger because the bin includes predictions with higher errors, making the model overconfident.

Looking at the PIT histogram (Fig. 14C), the Normal, SHASH, and CRPS methods fall relatively near the 0.1 line for all of the bins. There is a slight tendency for the predictions to fall into the central bins, which coincides with the models underconfidence for the predictions with the highest uncertainty. In contrast, the MC-dropout method shows that the majority of the predictions fall in the extremes of the distribution, further suggesting the model is overconfident. The RMSE values given in the labels are the RMSE errors between each bin and the expected 0.1 value. Here we see that while the model using the Normal distribution has the most uniform PIT, the SHASH and CRPS loss models perform similarly. In contrast, the RMSE for the MC-dropout model is large since the majority of the predictions fall in the outer bins.

Finally, the discard test (Fig. 14D) shows that for the Normal, SHASH, and CRPS methods the errors are reduced as the most uncertain predictions are removed, indicating that the uncertainty estimates are indeed well-calibrated. However, this is not the case for the MC-dropout method, where the error only slightly decreases. This adds further support to the previous two diagrams that for this application the MC-dropout model is not predicting well-calibrated, helpful uncertainty values for this application.

6. Demonstration of UQ and evaluation methods for a classification task

In this section we introduce the classification task (predicting convection), apply MC dropout and quantile regression to obtain uncertainty estimates, then evaluate the uncertainty estimates and show case studies.

a. Predicting convection from satellite imagery

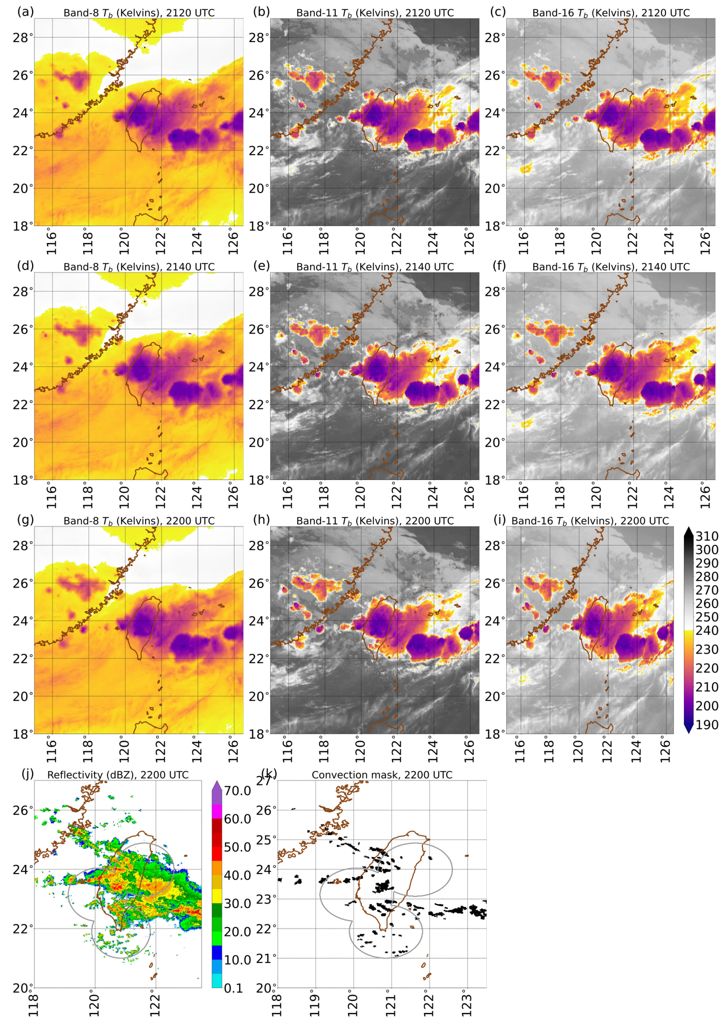

(a) Band-8 T_b (Kelvins), 2120 UTC
(b) Band-11 T_b (Kelvins), 2120 UTC
(c) Band-16 T_b (Kelvins), 2120 UTC
(d) Band-8 T_b (Kelvins), 2140 UTC
(e) Band-11 T_b (Kelvins), 2140 UTC
(f) Band-16 T_b (Kelvins), 2140 UTC
(g) Band-8 T_b (Kelvins), 2200 UTC
(h) Band-11 T_b (Kelvins), 2200 UTC
(i) Band-16 T_b (Kelvins), 2200 UTC
(j) Reflectivity (dBZ), 2200 UTC
(k) Convection mask, 2200 UTC

Figure 15: Input data for predicting convection at 2200 UTC 2 Jun 2017. Shown are three of the seven predictors: band 8 (6.25 μm), band 11 (8.6 μm), and band 16 (13.3 μm). [a-c] Predictors at lag time of 40 minutes; [d-f] predictors at lag time of 20 minutes; [g-i] predictors at lag time of 0 minutes. All predictors use the colour bar next to panel i. [j] Composite (column-maximum) radar reflectivity. [k] Convection mask. The black dots are pixels with true convection. Grey circles in panels h-i show the 100-km range ring around each radar.

The task is to predict the occurrence of thunderstorms (henceforth, "convection") at 1-hour lead time at each pixel in a grid. As predictors, we use gridded brightness temperatures from the Himawari-8 satellite at seven wavelengths and three lag times (0, 20, and 40 minutes before the forecast-issue time t_0), as shown in Fig. 15. The target is a binary convection mask at $t_0 + 1$ hour, created by applying an algorithm called Storm-labeling in 3 Dimensions (SL3D; Starzec et al. 2017) to radar data. Both the predictor and target variables are on a latitude-longitude grid with 0.0125° (~1.25-km) spacing. We train and evaluate the NNs only at pixels within 100 km of the nearest radar (grey circles in Figs. 15h-i), where radar coverage is sufficient to detect convection. See Lagerquist et al. (2021, henceforth L21) for complete details.

Our NN architecture is a U-net specially designed to predict gridded variables – here, the presence of convection at $t_0 + 1$ hour. Details on our particular architecture are in L21. The U-net in L21 is deterministic; in this paper we use either MC dropout or QR to make the U-net probabilistic. For MC dropout, we use the architecture in Fig. 16a. For QR, we replace the single output layer in Fig. 16a with $N + 1$ output layers, where N is the number of quantile levels estimated (Fig. 16b). We use Eq. 5 to prevent quantile-crossing, as represented schematically in the rightmost column of Fig. 16b.[4]

Because convection is a rare event (occurring at only 0.75% of pixels), we need to use an aggressive loss function (one that rewards true positives more than true negatives). Traditional loss functions, like the traditional quantile loss (Eq. 4), which reward true positives and true negatives equally, would result in a model that almost never predicts convection. Thus, we use a hybrid between the fractions skill score and traditional quantile loss, which we call the "aggressive quantile loss". See the Appendix for details.

b. UQ results

For this application we evaluate results from two different methods, MC dropout and QR. For MC dropout we tune three hyperparameters, specifically dropout rates for the last three layers. For QR we tune two hyperparameters: the set of quantile levels and the weight w in Eq. A2, which affects the importance of deterministic vs. probabilistic predictions in the loss function. Details are in Section 1 of the Supplemental Material. We run the MC-dropout models 100 times in inference

[4]For a regression problem, Eq. 5 alone is sufficient. For a classification problem like this one, quantile-based estimates $\hat{y}_i$ must range from $[0, 1]$, allowing them to be interpreted as probabilities. This is why, as shown in Fig. 16b, we apply the sigmoid activation function to the output of Eq. 5.

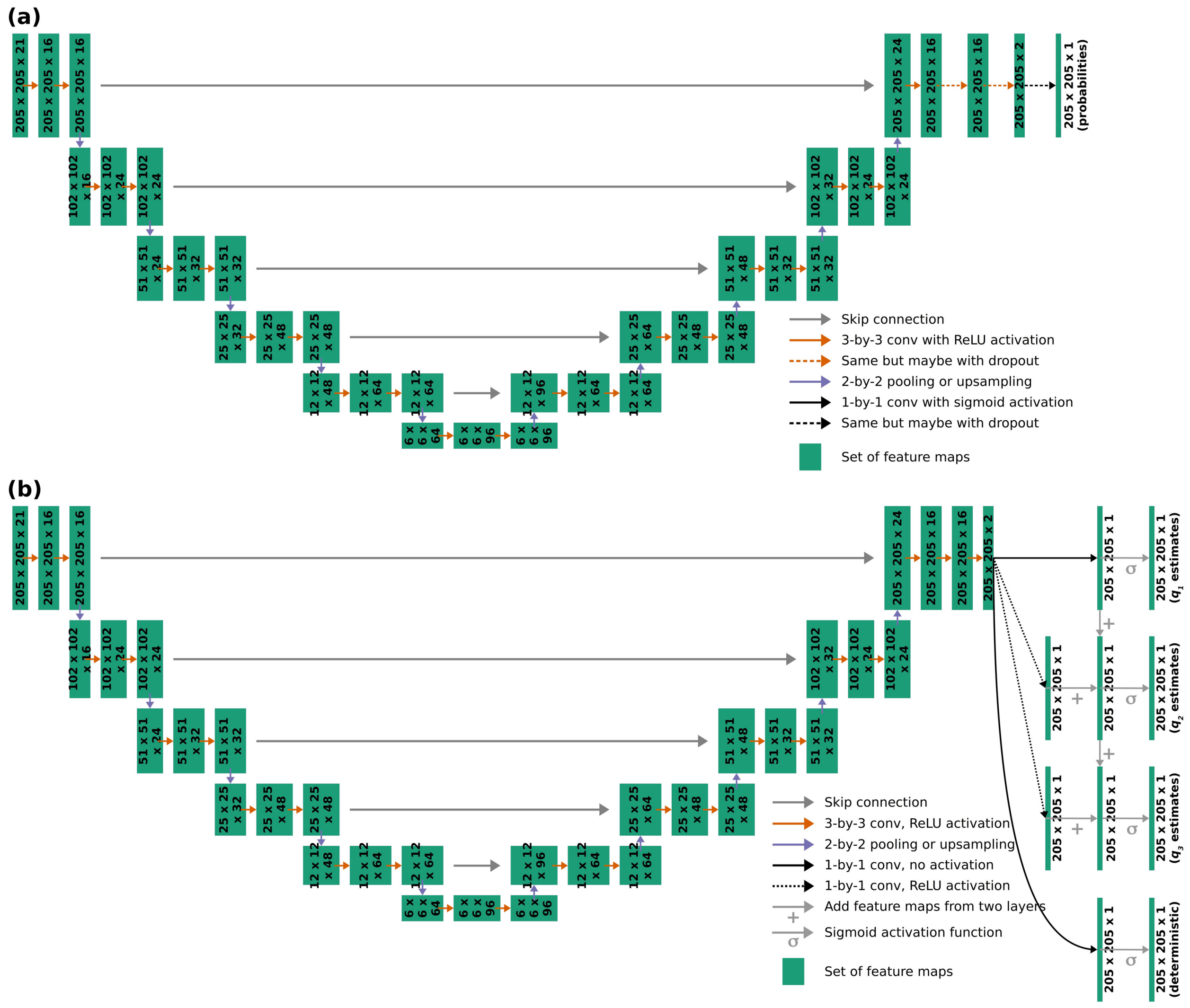

(a)
205 x 205 x 21
205 x 205 x 16
205 x 205 x 24
205 x 205 x 2
205 x 205 x 1 (probabilities)
102 x 102 x 16
102 x 102 x 24
102 x 102 x 32
51 x 51 x 24
51 x 51 x 32
51 x 51 x 48
25 x 25 x 32
25 x 25 x 48
25 x 25 x 64
12 x 12 x 48
12 x 12 x 64
12 x 12 x 96
6 x 6 x 64
6 x 6 x 96
Skip connection
3-by-3 conv with ReLU activation
Same but maybe with dropout
2-by-2 pooling or upsampling
1-by-1 conv with sigmoid activation
Same but maybe with dropout
Set of feature maps
(b)
205 x 205 x 1
205 x 205 x 1 (q_1 estimates)
205 x 205 x 1 (q_2 estimates)
205 x 205 x 1 (q_3 estimates)
205 x 205 x 1 (deterministic)
Skip connection
3-by-3 conv, ReLU activation
2-by-2 pooling or upsampling
1-by-1 conv, no activation
1-by-1 conv, ReLU activation
Add feature maps from two layers
Sigmoid activation function
Set of feature maps

Figure 16: U-net architecture for predicting convection from satellite imagery, using either [a] MC dropout or [b] quantile regression to estimate uncertainty. In each set of feature maps, the numbers are dimensions: $N_{\text{rows}} \times N_{\text{columns}} \times N_{\text{channels}}$. The 21 input channels are the raw predictor variables, *i.e.*, gridded brightness temperatures at 7 wavelengths and 3 lag times. In panel a, the last 3 convolutional layers are marked with dashed lines (either orange or black), indicating that these layers may include MC dropout. In panel b, to be brief, we assume that there are only 3 quantile levels to estimate. When there are more quantile levels (which is the case for every U-net involved in this study), the pattern on the right side of panel b repeats. Specifically, estimates for quantile level q_i are created by applying 1-by-1 convolution with ReLU to the feature maps marked "$205 \times 205 \times 2$," adding the result to pre-sigmoid estimates for quantile level q_{i-1}, then applying the sigmoid activation function. The sigmoid activation function constrains the final estimates to the range $[0, 1]$, so that they may be interpreted as probabilities of convection.

mode, yielding 100 estimates in each predictive distribution. We use the validation data to select hyperparameters and the independent testing data to show results for the selected models.

Monte Carlo dropout

As discussed in the Supplemental Material, as dropout rates increase, the uncertainty estimates improve but the mean predictions deteriorate. Hence, there is a trade-off between the quality of probabilistic and deterministic predictions. We judge that the best model has a dropout rate of 0.250 for the last layer, 0.125 for the second-last, and 0.375 for the third-last.

Fig. 17a shows the spread-skill plot for the best model. The model is overconfident (underspread) for all bins; this problem is typical for MC dropout, including atmospheric-science applications (Scher and Messori 2021; Clare et al. 2021; Garg et al. 2022). Also, the model rarely produces spread values > 0.04, as shown in the histogram in Fig. 17a. The skewed histogram is explained by the inset in Fig. 17a: model spread increases almost linearly with convection frequency (the mean target value), and since convection is a rare event, we should therefore expect high model spread to be a rare event.

Fig. 17b shows the discard test for the best model. Model error decreases every time the discard fraction increases (18 of 19 times), yielding an MF of 94.74%. Thus, the ranking quality of the model's uncertainty estimates is high. As shown in the Fig. 17b inset, event frequency decreases from 0.75% to 0.2% after 10% of the highest-uncertainty predictions are discarded. In other words, most convection is associated with very high uncertainty, consistent with the Fig. 17a inset.

Quantile regression (QR)

As discussed in the Supplemental Material, results of the hyperparameter experiment are noisy. We judge that the best model has 17 quantile levels and a weight of 6 ($w = 6$ in Eq. A2). To compute the model spread, defined as the standard deviation of the predictive distribution, we use Eq. 15 of Wan et al. (2014).

The spread-skill plot for the best model shows that it is almost perfectly calibrated for spread bins ≤ 0.06 and underconfident (overspread) for spread bins > 0.06 (Fig. 17c). However, spread rarely exceeds 0.06 (only for 22.7% of examples, as shown by the histogram), so the model is generally well calibrated. The spread-skill plots reveal two advantages of the QR model (Fig. 17c) over the MC-dropout model (Fig. 17a). First, the QR model is better-calibrated overall, which manifests in a substantially lower SSREL (0.025 versus 0.035); the difference is significant at the 95% confidence level[5] Second, the QR model's spread histogram is less skewed, *i.e.*, the QR

[5]Determined by a one-sided paired bootstrap test with 1000 iterations.

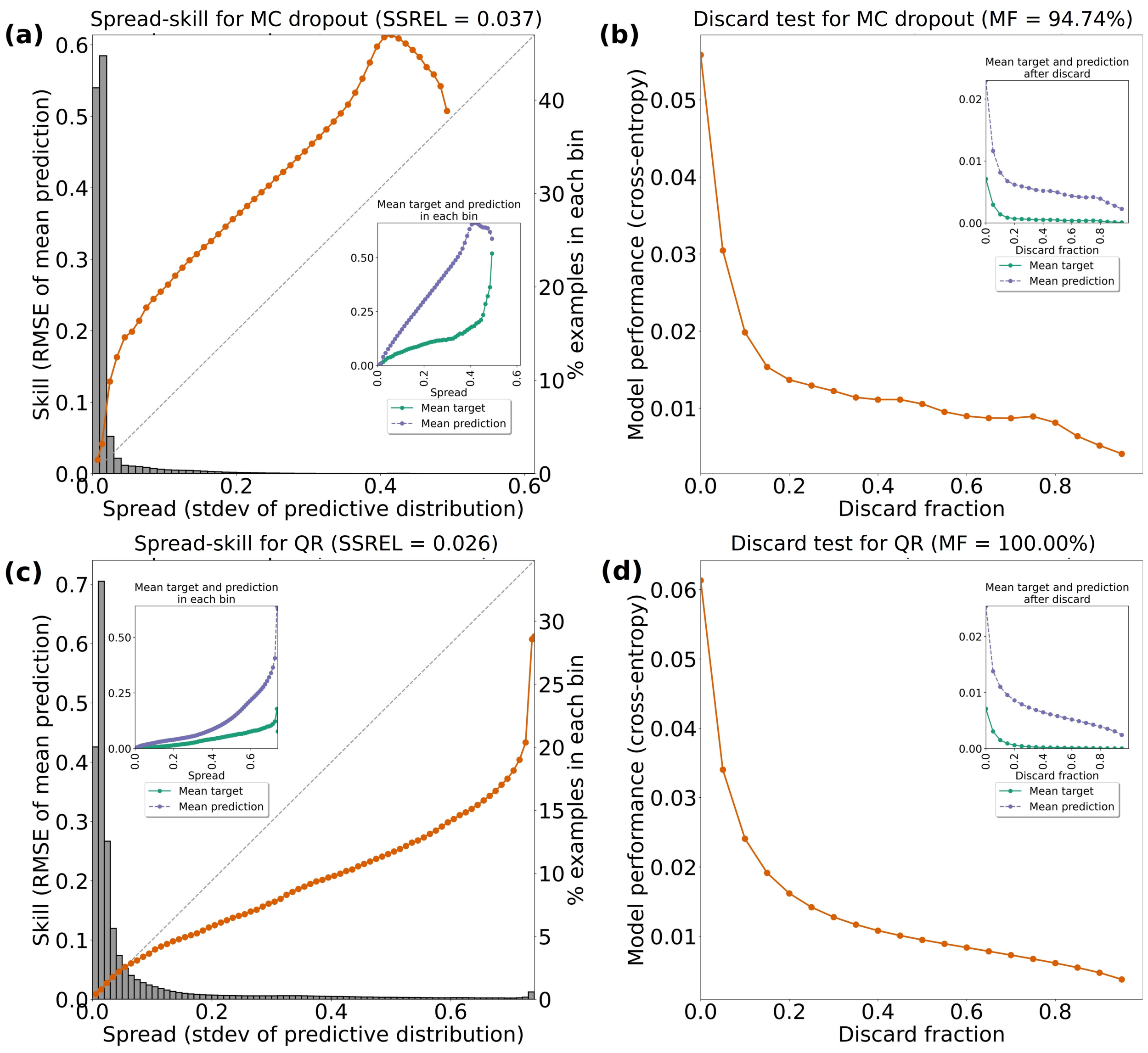


Figure 17: [a] Spread-skill plot and [b] discard test, measured on testing data, when using MC dropout to predict convection. [c] Spread-skill plot and [d] discard test, measured on testing data, when using quantile regression to predict convection. “SSREL” is spread-skill reliability, and “MF” is monotonicity fraction. The histogram in the spread-skill plot shows the percentage of testing examples in each bin of spread values; the bins have a spacing of 0.01.

model produces high spread values (> 0.04) more often. However, the QR model is not much better-calibrated at these high spread values. As mentioned above, the overconfidence problem for MC dropout (which occurs for all 125 models we trained) is well documented in the literature. However, to our knowledge, the underconfidence problem for QR (which occurs for all 90 models we trained) has not been noted previously. Thus, we are unsure whether this is a problem with QR in general or specific to our application.

Fig. 17d shows the discard test for the best QR model. Error decreases monotonically as high-uncertainty predictions are discarded, yielding an MF of 100%, compared to 94.74% for the MC-dropout model. Also, error decreases more smoothly with discard fraction for the QR model than for the MC-dropout model (Fig. 17b).

CASE STUDY

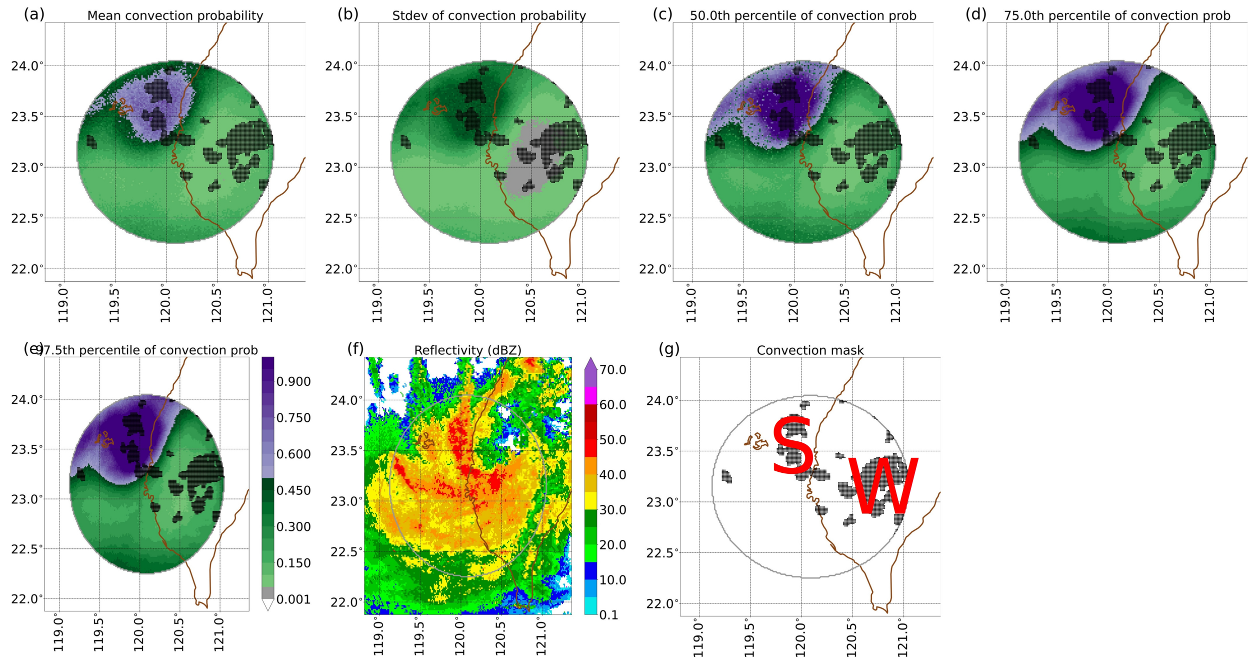

(a) Mean convection probability
(b) Stdev of convection probability
(c) 50.0th percentile of convection prob
(d) 75.0th percentile of convection prob
(e) 97.5th percentile of convection prob
(f) Reflectivity (dBZ)
(g) Convection mask
S
W

Figure 18: Case study for MC-dropout model, during Tropical Depression Luis. All data (predictions, radar reflectivity, and convection mask) are valid at 0830 UTC 23 Aug 2018. The predictions were made with 1-hour lead time (initialized at 0730 UTC). [a] Mean convection probability, [b] spread, [c] median convection probability; [d] 75th-percentile convection probability; [e] 97.5th-percentile convection probability; [f] composite (column-maximum) radar reflectivity; [g] true convection mask, with black dots showing convective pixels. "S" indicates an area of strong convection west of the center of Luis, while "W" indicates an area of weak convection east of the center.

Figs. 18-19 show a case study created by applying the best MC-dropout model and best QR model to selected time steps in the testing data. Each figure summarizes the predictive distribution with five numbers: the mean, standard deviation, and three percentiles of convection probability. We do not show percentiles below the 50th, because estimates corresponding to these percentiles

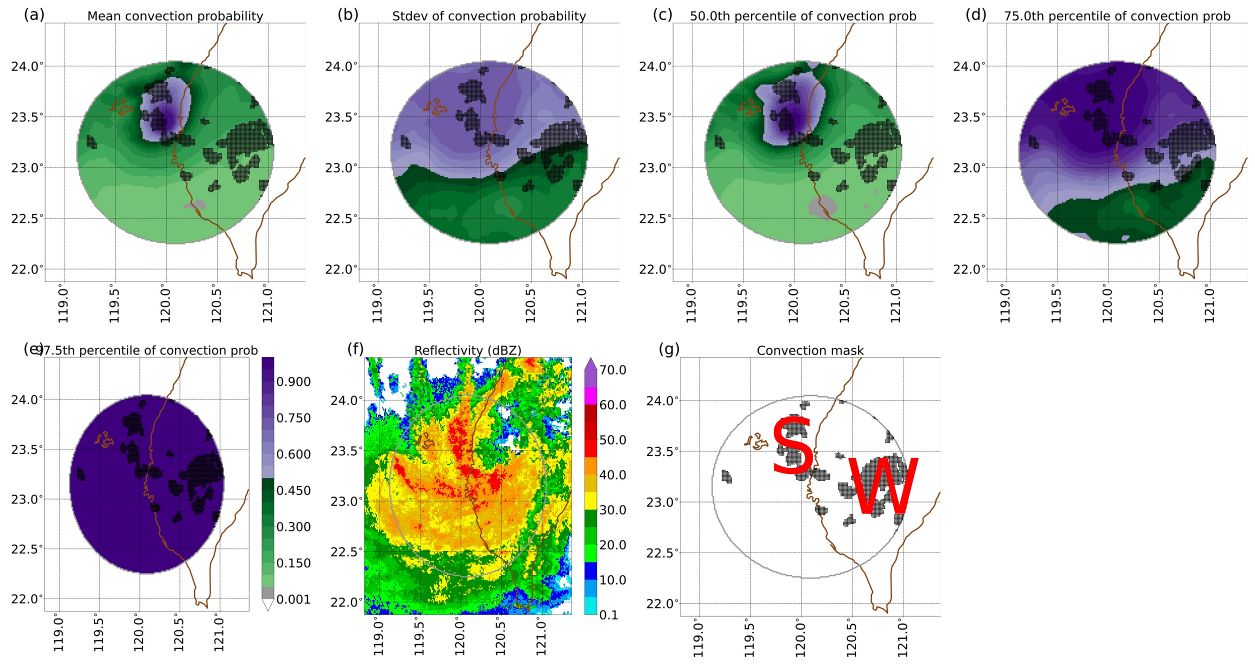


Figure 19: Same as Fig. 18 but for QR model.

are usually very small; most of the variation in the predictive distribution is between the 50th and 97.5th percentiles.

Figs. 18 and 19 show a case during Tropical Depression Luis. This case features two large areas of convection: strong ("S") and weak ("W"). We make six observations from these figures. First, in terms of the mean and any percentile below the 97.5th, both models produce higher probabilities for strong convection than for weak convection (panels a and c-e), which is a desired property[6]. Second, both models are more uncertain for the strong convection than the weak convection (panel b). This is not a desired property, because weak convection (borderline cases) is more difficult to identify and should have higher uncertainty. The third conclusion counteracts the second: the standard deviation is not the full story on uncertainty, and sometimes it is necessary to look at the full predictive distribution. The MC-dropout model has almost no difference between the 50th- and 97.5th-percentile estimates in area W (Fig. 18c-e), consistent with the low standard deviations (Fig. 18b). However, the QR model has large differences between the 50th- and 97.5th-percentile estimates in area W (Fig. 19c-e), despite the standard deviations here being smaller than elsewhere in the domain (Fig. 19b). Fourth, probability maps produced by the MC-dropout model contain

[6]For the 97.5th percentile the MC-dropout model also produces higher probabilities for strong convection (Fig. 18e), but the QR model produces nearly equal probabilities, around 100%, over the full domain (Fig. 19e).

checkerboard artifacts (Fig. 18a-c); these are caused by using dropout in the last layer, which sets some probabilities to zero. Fifth, the overconfidence (underspread) problem with the MC-dropout model is obvious in area W, where there is almost no difference between the 50^{th}- and 97.5^{th}-percentile estimates (Fig. 18c-e). Sixth, the underconfidence (overspread) problem with the QR model is also obvious, where the 97.5^{th}-percentile estimate is essentially 100% everywhere in the domain (Fig. 19e).

For an additional case study during the winter, see Section 2 of the Supplemental Material.

7. Discussion and Conclusions

Uncertainty predictions are an important tool for understanding ML models, as these estimates can provide valuable information as to when to trust the model predictions. For tasks that involve critical decisions, these estimates are invaluable. In recent years, there has been a surge in research focusing on developing different methods for quantifying uncertainty in ML models; however, it is essential that along with developing the methods themselves there are helpful and useful ways to evaluate uncertainty predictions.

To this end, we have demonstrated four different evaluation techniques for two different applications, a binary classification task and a regression task. To summarize our thoughts and findings:

- The **Reliability Diagram** provides an excellent way to evaluate the central predictions for both binary and regression tasks. By providing a comparison of the predicted and observed values along with their histograms, this single diagram reveals where the predictions are performing poorly and indicates any biases. In addition to providing a visual for the prediction quality across the entire range of observed values, it also provides the Brier score for binary predictions.

- The **Spread-Skill Plot** provides a single, comprehensive figure for how the model error compares to the model uncertainty prediction. Ideally, the uncertainty should match the error, and this plot is an easy way to visualize this information and show whether or not the model is well-calibrated. Additionally, by including the histogram of spreads, this plot also provides information regarding the performance of the model. Since a low error is desirable, we want most of the counts to be in the bins with the lowest spread and lowest error, and this can quickly be seen from the histogram. Additionally, numerous different models can be plotted on a single diagram, making it helpful in model comparison tasks.

- The **PIT Histogram** provides information as to whether the uncertainty spread is appropriate for regression tasks, producing a quick metric for whether a model is over- or underconfident. However, the information from this plot can also be gleaned from the spread-skill diagram, where points above the 1:1 line indicate the model is underconfident and points below the 1:1 line indicate the model is overconfident. The spread-skill diagram also provides information regarding whether the mismatched uncertainties correspond with low- or high-error cases; however, the PIT histogram is useful for quickly determining if the model uncertainty estimates appropriately match the model error.

- The **Discard Test** provides information on whether the overall error is correlated with the uncertainty. For most tasks we can improve model performance by removing the cases with the highest uncertainty. This figure quickly identifies if this is the case. If this is not the case, then one of two scenarios may be occurring. The first possibility is that the model uncertainty is not well-calibrated, thus removing the uncertain predictions is not helpful in reducing the error. The second possibility is that removing uncertain predictions is not appropriate because the true data spread is constant and there are no cases that can be more-reliably predicted. Although the information in this figure is included in the spread-skill plot, this figure provides a quick and easy-to-interpret check on whether the most uncertain predictions are indeed performing the worst.

  The information in the discard test can be deduced from the spread-skill plot with some analysis. If the spread-skill line lies on the 1:1 line and the counts are highest for the lower errors, then removing the predictions with the highest spread will remove the cases with the higher error. In contrast, if the spread-skill diagram indicates the model spread is not well-matched to the skill (*e.g.*, a flat spread-skill line) then prediction skill is not correlated with the uncertainty spread and removing the predictions with high uncertainty will not reduce the overall error. In a separate scenario, if the bins used in the spread-skill diagram are very close together or if the counts are localized to minimal spread bins, this is an indication that there are no cases that have higher uncertainty than others.

In evaluating model performance, the above metrics are designed for circumstances where distribution parameters, such as the mean and standard deviation, are appropriate. It should be kept in mind that evaluating the broad-scale statistics of the uncertainty may not be appropriate for all tasks and models. For tasks where the target may have more than one primary outcome, such as identifying the likelihood of precipitation or ensemble UQ methods, these metrics may be misleading. In these circumstances, the model may appear to be performing poorly even if all of the ensembles are representative of the data, particularly in the reliability diagram. Additionally, the discard test may show amplified improvements for these circumstances. For example, if there is more than one probable target for some cases in a data set, these cases will always have a higher error with the simple discard test used here and will be removed, even though they may be well-captured with an ensemble-based uncertainty method. In these situations, we have found that the best way to evaluate overall model performance is either to analyze figures that include the testing data and all of the model ensemble predictions or to evaluate cases on an individual basis; however, even with ensemble approaches, the metrics here still provide useful information as long as they are interpreted correctly.

Although it was not a primary goal of this study, evaluating the central predictions and uncertainty estimates on the two tasks here allowed us to compare the different uncertainty methods. During our analysis, we came to some findings on the different UQ methods that may be of use to the community.

- QR, although typically used for regression tasks, can be useful for classification tasks. Compared to MC dropout on the convection application, QR produces a better spread-skill plot, a better discard test, and more useful uncertainty estimates in the case studies shown.

- Parametric regression methods (*i.e.,* predicting normal or SHASH distribution parameters) and using the CRPS as the loss function to create ensemble predictions are useful approaches to predicting well-calibrated uncertainties for regression tasks.

- QR, parametric regressions, and CRPS loss perform well for the majority of cases, but all appear to be underconfident for high-error cases.

- MC dropout does not appear to work well, particularly for regression tasks. Rather than creating reliable uncertainty estimates, this method appears to predict uncertainties that are overconfident and not well-calibrated.

Acknowledgments. This work was funded in part by NSF AI institute grant ICER-2019758 (MM, IE), NSF HDR grant OAC-1934668 (IE), and NOAA Grant NA19OAR4320073 (all authors). We thank Dr. Steven Brey for providing his expertise on the CRPS method and making his CRPS-Net code available.

Data availability statement. The Jupyter notebooks referenced throughout this article are available in a GitHub repository, which you can find at `https://github.com/thunderhoser/cira\_uq4ml`. Additionally, we have created a frozen release of the notebooks for this article at `https://doi.org/10.5281/zenodo.6915051`.

APPENDIX

Aggressive quantile loss for predicting convection

The fractions skill score (FSS; Roberts and Lean 2008) rewards true positives more than true negatives, making it well suited for rare-event prediction. The U-net for QR (Figure 16b) has $N+1$ outputs, where N is the number of quantile levels estimated. The last output is the deterministic prediction, and its loss function is the pixelwise FSS (*i.e.*, FSS without a neighbourhood filter). The i^{th} output ($i \leq N$) is the estimate for quantile level q_i, and its loss function is a hybrid between the traditional quantile loss (Eq. 4) and the pixelwise FSS:

$$\mathcal{L}_i = \begin{cases} (1-q_i)\frac{(y-\hat{y}_{q_i})^2}{y^2+\hat{y}_{q_i}^2} & , \quad y \leq \hat{y}_{q_i}; \\ q_i \frac{(y-\hat{y}_{q_i})^2}{y^2+\hat{y}_{q_i}^2} & , \quad y > \hat{y}_{q_i}. \end{cases} \tag{A1}$$

y is the true value from the convection mask, and $\hat{y}_{q_i}$ is the estimate for quantile level q_i, ranging from $[0,1]$. You might ask: why use the FSS without a neighbourhood filter? After all, a key benefit of the FSS is that it uses a neighbourhood filter to solve the double-penalty problem, where a model is punished too harshly for a small (*e.g.*, 1-pixel) offset between the predicted and observed event. The answer is that (a) convection is a rare event, occurring at only 0.75% of pixels on average; (b) to obtain a U-net that predicts substantial probabilities for a rare event, it is necessary to use a loss function that rewards true positives more than true negatives; (c) the FSS, even without a neighbourhood filter, has this desired property; (d) including a filter in the loss function requires the inclusion of a filter in the UQ evaluation metrics, which is more complication than we wanted. We call the loss function in Eq. A1 the "aggressive quantile loss".

To compute the total loss for the i^{th} output ($i \leq N$), we average Eq. A1 over all pixels and valid times, yielding $\overline{\mathcal{L}_i}$. To compute the total loss for the U-net, we use the equation:

$$\overline{\mathcal{L}_{\text{model}}} = w\overline{\mathcal{L}_{\text{deterministic}}} + \frac{1}{N}\sum_{i=1}^{N}\overline{\mathcal{L}_i}, \tag{A2}$$

where $\overline{\mathcal{L}_{\text{deterministic}}}$ is the loss for the deterministic prediction (pixelwise FSS) and $w > 1$ is a user-selected weight. When there are many quantile levels (*i.e.*, N is large), this weight emphasizes the deterministic predictions, ensuring that both deterministic and probabilistic predictions are skillful.

References

Barnes, E., and R. Barnes, 2021: Controlled abstention neural networks for identifying skillful predictions for regression problems. *Journal of Advances in Modeling Earth Systems*, **13 (12)**, e2021MS002 575, URL https://doi.org/10.1029/2021MS002575.

Barnes, E., R. Barnes, and N. Gordillo, 2021: Adding uncertainty to neural network regression tasks in the geosciences. *arXiv e-prints*, **2109 (07250)**, URL https://arxiv.org/abs/2109.07250.

Benjamin, S., and Coauthors, 2016: A North American hourly assimilation and model forecast cycle: The Rapid Refresh. *Monthly Weather Review*, **144 (4)**, 1669–1694, URL https://doi.org/10.1175/MWR-D-15-0242.1.

Bihlo, A., 2021: A generative adversarial network approach to (ensemble) weather prediction. *Neural Networks*, **139**, 1–16, URL https://doi.org/10.1016/j.neunet.2021.02.003.

Brey, S., 2021: Ensemble. GitHub, URL https://github.com/TheClimateCorporation/ensemble.

Cannon, A., 2011: Quantile regression neural networks: Implementation in R and application to precipitation downscaling. *Computers and Geosciences*, **37 (9)**, 1277–1284, URL https://doi.org/10.1016/j.cageo.2010.07.005.

Chapman, W., L. D. Monache, S. Alessandrini, A. Subramanian, F. Ralph, S. Xie, S. Lerch, and N. Hayatbini, 2022: Probabilistic predictions from deterministic atmospheric river forecasts with deep learning. *Monthly Weather Review*, **150 (1)**, 215–234, URL https://doi.org/10.1175/MWR-D-21-0106.1.

Clare, M., O. Jamil, and C. Morcrette, 2021: Combining distribution-based neural networks to predict weather forecast probabilities. *Quarterly Journal of the Royal Meteorological Society*, **147 (741)**, 4337–4357, URL https://doi.org/10.1002/qj.4180.

Delle Monache, L., F. Eckel, D. Rife, B. Nagarajan, and K. Searight, 2013: Probabilistic weather prediction with an analog ensemble. *Monthly Weather Review*, **141 (10)**, 3498–3516, URL https://doi.org/10.1175/MWR-D-12-00281.1.

Dillon, J. V., and Coauthors, 2017: Tensorflow distributions. arXiv, URL https://arxiv.org/abs/1711.10604, https://doi.org/10.48550/ARXIV.1711.10604.

Dürr, O., B. Sick, and E. Murina, 2020: *Probabilistic deep learning: With python, keras and tensorflow probability*. Manning Publications.

Fukushima, K., and S. Miyake, 2022: Incorporating uncertainty into a regression neural network enables identification of decadal state-dependent predictability. *Geophysical Research Letters*, **submitted**, URL https://doi.org/10.1002/essoar.10510836.1.

Gal, Y., and Z. Ghahramani, 2016: Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. *International Conference on Machine Learning*, **48**, 1050–1059, URL http://proceedings.mlr.press/v48/gal16.html?ref=https://githubhelp.com.

Garg, S., S. Rasp, and N. Thuerey, 2022: WeatherBench Probability: A benchmark dataset for probabilistic medium-range weather forecasting along with deep learning baseline models. *arXiv e-prints*, **2205 (00865)**, URL https://arxiv.org/abs/2205.00865.

Gneiting, T., and A. Raftery, 2007: Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, **102 (477)**, 359–378, URL https://doi.org/10.1198/016214506000001437.

Gneiting, T., A. Raftery, A. Westveld, and T. Goldman, 2005: Calibrated probabilistic forecasting using ensemble model output statistics and minimum CRPS estimation. *Monthly Weather Review*, **133 (5)**, 1098–1118, URL https://doi.org/10.1175/MWR2904.1.

Grönquist, P., C. Yao, T. Ben-Nun, N. Dryden, P. Dueben, S. Li, and T. Hoefler, 2021: Deep learning for post-processing ensemble weather forecasts. *Philosophical Transactions of the Royal Society A*, **379 (2194)**, 20200 092, URL https://doi.org/10.1098/rsta.2020.0092.

Hamill, T., 2001: Interpretation of rank histograms for verifying ensemble forecasts. *Monthly Weather Review*, **129 (3)**, 550–560, URL https://doi.org/10.1175/1520-0493(2001)129%3C0550:IORHFV%3E2.0.CO;2.

Hersbach, H., 2000: Decomposition of the continuous ranked probability score for ensemble prediction systems. *Weather and Forecasting*, **15 (5)**, 559–570, URL https://doi.org/10.1175/1520-0434(2000)015%3C0559:DOTCRP%3E2.0.CO;2.

Hinton, G., N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, 2012: Improving neural networks by preventing co-adaptation of feature detectors. *arXiv e-prints*, **1207 (0580)**, URL https://arxiv.org/abs/1207.0580.

Hsu, W., and A. Murphy, 1986: The attributes diagram: A geometrical framework for assessing the quality of probability forecasts. *International Journal of Forecasting*, **2 (3)**, 285–293, URL https://doi.org/10.1016/0169-2070(86)90048-8.

Jospin, L., H. Laga, F. Boussaid, W. Buntine, and M. Bennamoun, 2022: Hands-on Bayesian neural networks – A tutorial for deep learning users. *IEEE Computational Intelligence Magazine*, **17 (2)**, 29–48, URL https://doi.org/10.1109/MCI.2022.3155327.

Klotz, D., F. Kratzert, M. Gauch, A. Keefe Sampson, J. Brandstetter, G. Klambauer, S. Hochreiter, and G. Nearing, 2021: Uncertainty estimation with deep learning for rainfall–runoff modelling. *Hydrology and Earth System Sciences Discussions*, 1–32.

Lagerquist, R., J. Stewart, I. Ebert-Uphoff, and C. Kumler, 2021: Using deep learning to nowcast the spatial coverage of convection from Himawari-8 satellite data. *Monthly Weather Review*, **149 (12)**, 3897–3921, URL https://doi.org/10.1175/MWR-D-21-0096.1.

Matheson, J., and R. Winkler, 1976: Scoring rules for continuous probability distributions. *Management Science*, **22 (10)**, 1087–1096, URL https://doi.org/10.1287/mnsc.22.10.1087.

Nair, V., and G. Hinton, 2010: Rectified linear units improve restricted Boltzmann machines. *International Conference on Machine Learning*, Haifa, Israel, International Machine Learning Society, URL https://openreview.net/forum?id=rkb15iZdZB.

Orescanin, M., V. Petković, S. Powell, B. Marsh, and S. Heslin, 2021: Bayesian deep learning for passive microwave precipitation type detection. *IEEE Geoscience and Remote Sensing Letters*, **19**, 1–5, URL https://doi.org/10.1109/LGRS.2021.3090743.

Ortiz, P., M. Orescanin, V. Petković, S. Powell, and B. Marsh, 2022: Decomposing satellite-based classification uncertainties in large earth science datasets. *IEEE Transactions on Geoscience and Remote Sensing*, **60**, 1–11, URL https://doi.org/10.1109/TGRS.2022.3152516.

Rasp, S., and S. Lerch, 2018: Neural networks for postprocessing ensemble weather forecasts. *Monthly Weather Review*, **146 (11)**, 3885–3900, URL https://doi.org/10.1175/MWR-D-18-0187.1.

Roberts, N., and H. Lean, 2008: Scale-selective verification of rainfall accumulations from high-resolution forecasts of convective events. *Monthly Weather Review*, **136 (1)**, 78–97, URL https://doi.org/10.1175/2007MWR2123.1.

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

Salama, K., 2021: Probabilistic Bayesian Neural Networks. Keras, URL https://keras.io/examples/keras_recipes/bayesian_neural_networks/.

Sato, S., M. Takanashi, K. Indo, N. Nishihara, H. Ichikawa, and H. Watanabe, 2021: Two-Stage probabilistic short-term wind power prediction using neural network with MC dropout and control information. *International Exhibition and Conference for Power Electronics, Intelligent Motion, Renewable Energy and Energy Management*, 1–8, URL https://ieeexplore.ieee.org/abstract/document/9472407.

Scher, S., and G. Messori, 2021: Ensemble methods for neural network-based weather forecasts. *ournal of Advances in Modeling Earth Systems*, **13 (2)**, URL https://doi.org/10.1029/2020MS002331.

Schmit, T., P. Griffith, M. Gunshor, J. Daniels, S. Goodman, and W. Lebair, 2017: A closer look at the ABI on the GOES-R series. *Bulletin of the American Meteorological Society*, **98 (4)**, 681–698, URL https://doi.org/10.1175/BAMS-D-15-00230.1.

Seoh, R., 2020: Qualitative analysis of monte carlo dropout. *arXiv preprint arXiv:2007.01720*.

Starzec, M., C. Hometer, and G. Mullendore, 2017: Storm Labeling in Three Dimensions (SL3D): A volumetric radar echo and dual-polarization updraft classification algorithm. *Monthly Weather Review*, **145 (3)**, 1127–1145, URL https://doi.org/10.1175/MWR-D-16-0089.1.

Stock, J., 2021: Using machine learning to improve vertical profiles of temperature and moisture for severe weather nowcasting. M.S. thesis, Computer Science, Colorado State University, URL https://hdl.handle.net/10217/233704.

Székely, G., and M. Rizzo, 2005: A new test for multivariate normality. *Journal of Multivariate Analysis*, **93 (1)**, 58–80, URL https://doi.org/10.1016/j.jmva.2003.12.002.

Taylor, J., 2000: A quantile regression neural network approach to estimating the conditional density of multiperiod returns. *Journal of Forecasting*, **19 (4)**, 299–311, URL https://doi.org/10.1002/1099-131X(200007)19:4%3C299::AID-FOR775%3E3.0.CO;2-V.

Wan, X., W. Wang, J. Liu, and T. Tong, 2014: Estimating the sample mean and standard deviation from the sample size, median, range and/or interquartile range. *BMC Medical Research Methodology*, **14 (1)**, 1–13, URL https://doi.org/10.1186/1471-2288-14-135.

Yagli, G., D. Yang, and D. Srinivasan, 2022: Ensemble solar forecasting and post-processing using dropout neural network and information from neighboring satellite pixels. *Renewable and Sustainable Energy Reviews*, **155**, 111 909, URL https://doi.org/10.1016/j.rser.2021.111909.

Yu, Y., X. Han, M. Yang, and J. Yang, 2020: Probabilistic prediction of regional wind power based on spatiotemporal quantile regression. *IEEE Transactions on Industry Applications*, **56 (6)**, 6117–6127, URL https://doi.org/10.1109/TIA.2020.2992945.

Supplemental Material:
Creating and evaluating uncertainty estimates with neural networks for environmental-science applications

This article has been submitted to the AMS journal *Artificial Intelligence for the Earth Systems*.

1. Hyperparameter experiments for classification task

To reiterate, the classification task is predicting convection from satellite imagery. We perform one hyperparameter experiment per UQ method used for this task: MC dropout and QR.

a. Monte Carlo dropout

We attempt five dropout rates per layer (0, 0.125, 0.25, 0.375, 0.5) and use a grid search to attempt all $5^3 = 125$ possible combinations. To limit the size of the hyperparameter experiment, we do not attempt dropout for shallower layers (before the last three). All other hyperparameters, which we do not tune for this paper, are documented in L21. As in L21, we use the year 2016 for training, 2017 for validation, and 2018 for testing. We run the U-nets 100 times with MC dropout, yielding 100 estimates in each predictive distribution (*i.e.*, for each pixel at each valid time). To evaluate deterministic predictions (the mean of the distribution), we use the pixelwise FSS; to evaluate probabilistic predictions (the full distribution), we use SSREL (Eq. 7 in main body) and monotonicity fraction from the discard test (MF; Eq. 9 in main body).

Figs. S1-S3 show all three scores, computed on the validation data, for all 125 models. From these figures we make three general conclusions. First, MF improves (increases) as dropout rate for the last layer increases. Second, SSREL improves (decreases) as dropout rates for the last two layers increase. Third, FSS deteriorates (decreases) as dropout rates for the last two layers increase. Hence, there is a trade-off between the quality of probabilistic predictions (measured by

MF and SSREL) and deterministic predictions (measured by FSS). Based on Supplemental Figs. S1-S3, we judge that the best model has a dropout rate of 0.250 for the last layer, 0.125 for the second-last, and 0.375 for the third-last. This model achieves the 35th-best FSS (0.222), 44th-best SSREL (0.037), and 65th-best MF (0.895).

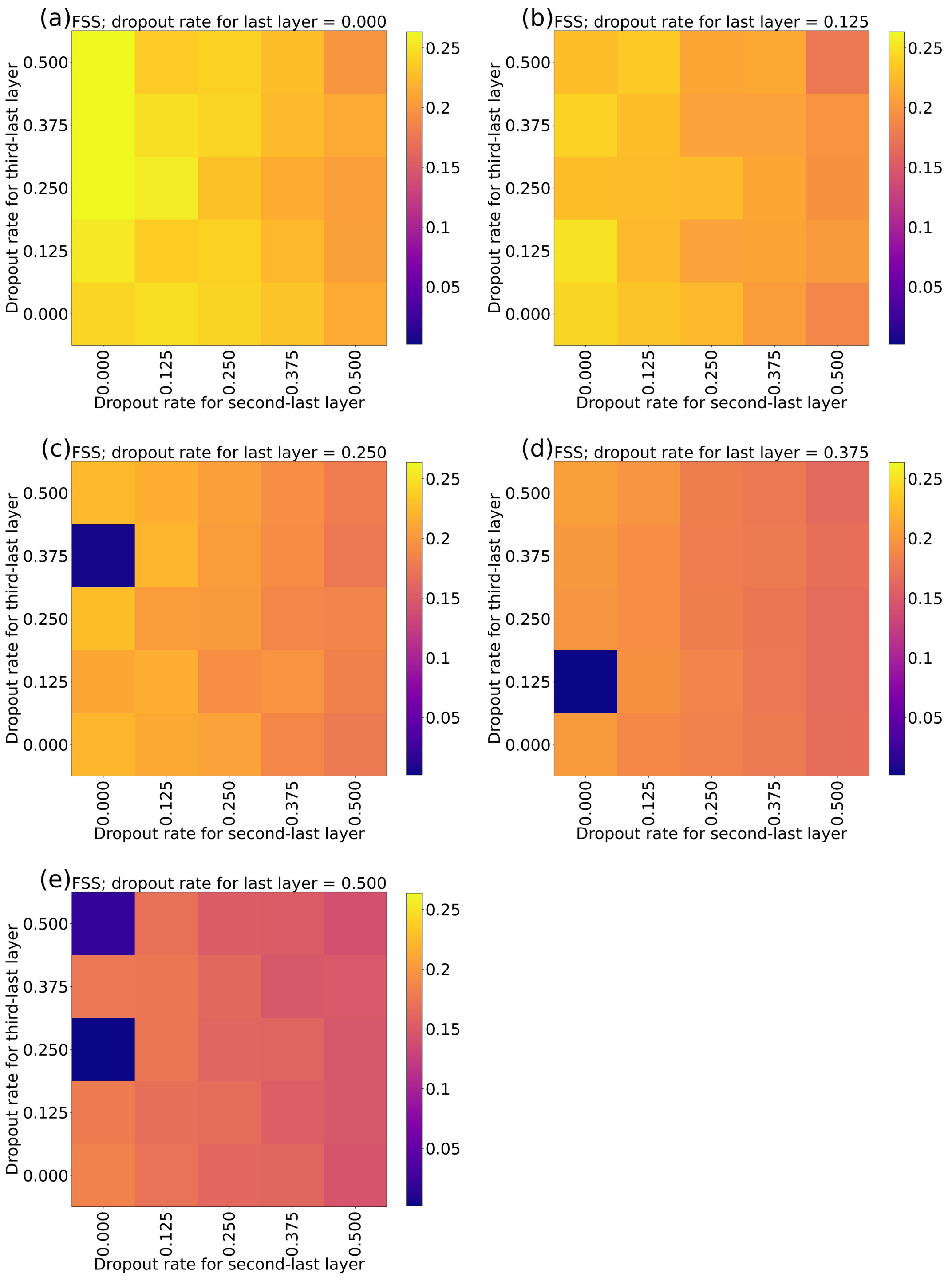


Figure S1: Pixelwise (1-by-1) fractions skill score (FSS), measured on validation data, when using MC dropout to predict convection. Each panel corresponds to a different dropout rate for the last layer, which is one hyperparameter in the experiment. The other hyperparameters are dropout rate for the second-last layer (x-axis in each panel) and third-last layer (y-axis in each panel).

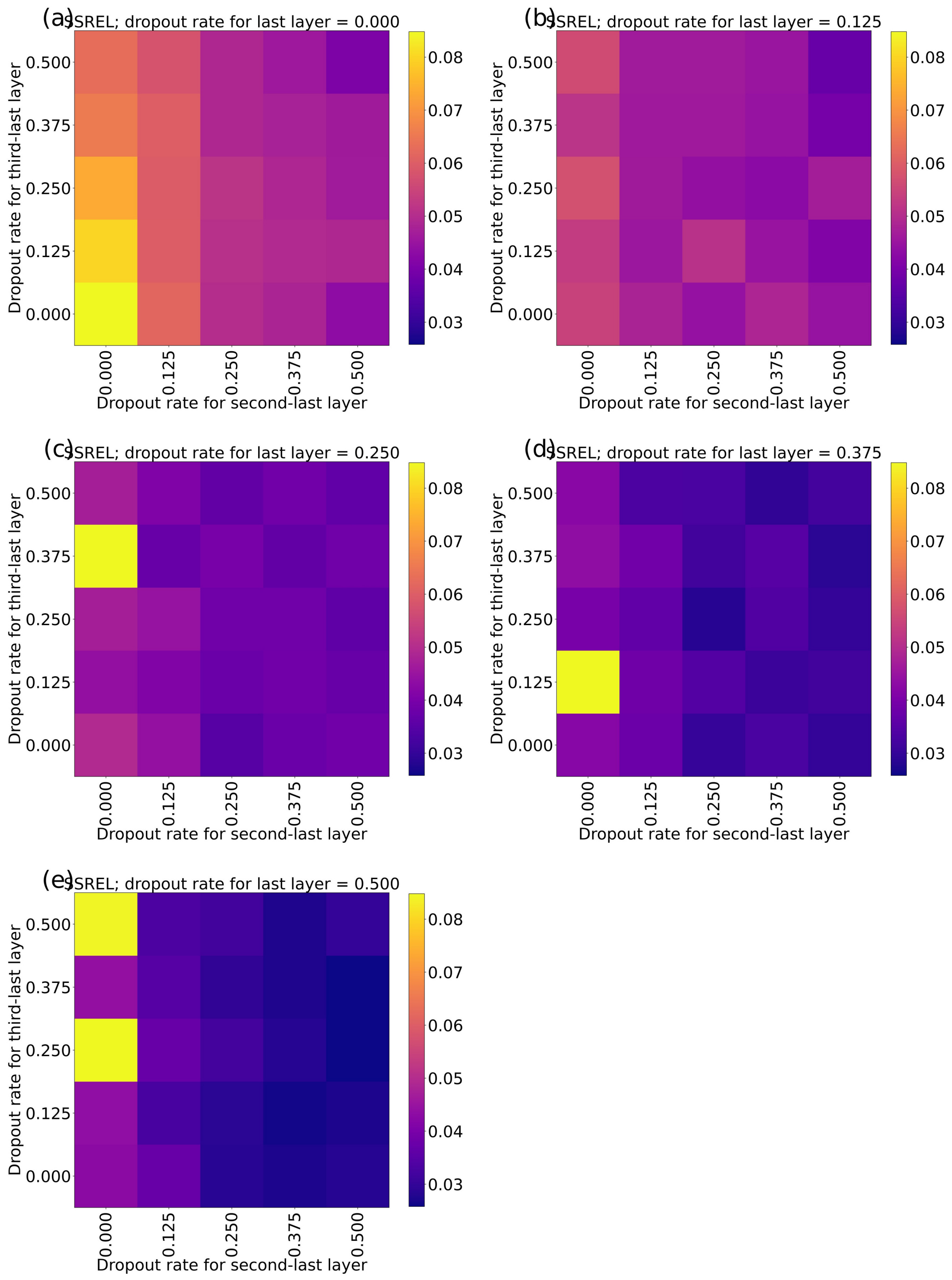


Figure S2: Spread-skill reliability (SSREL), measured on validation data, when using MC dropout to predict convection. Formatting is the same as in Figure S1.

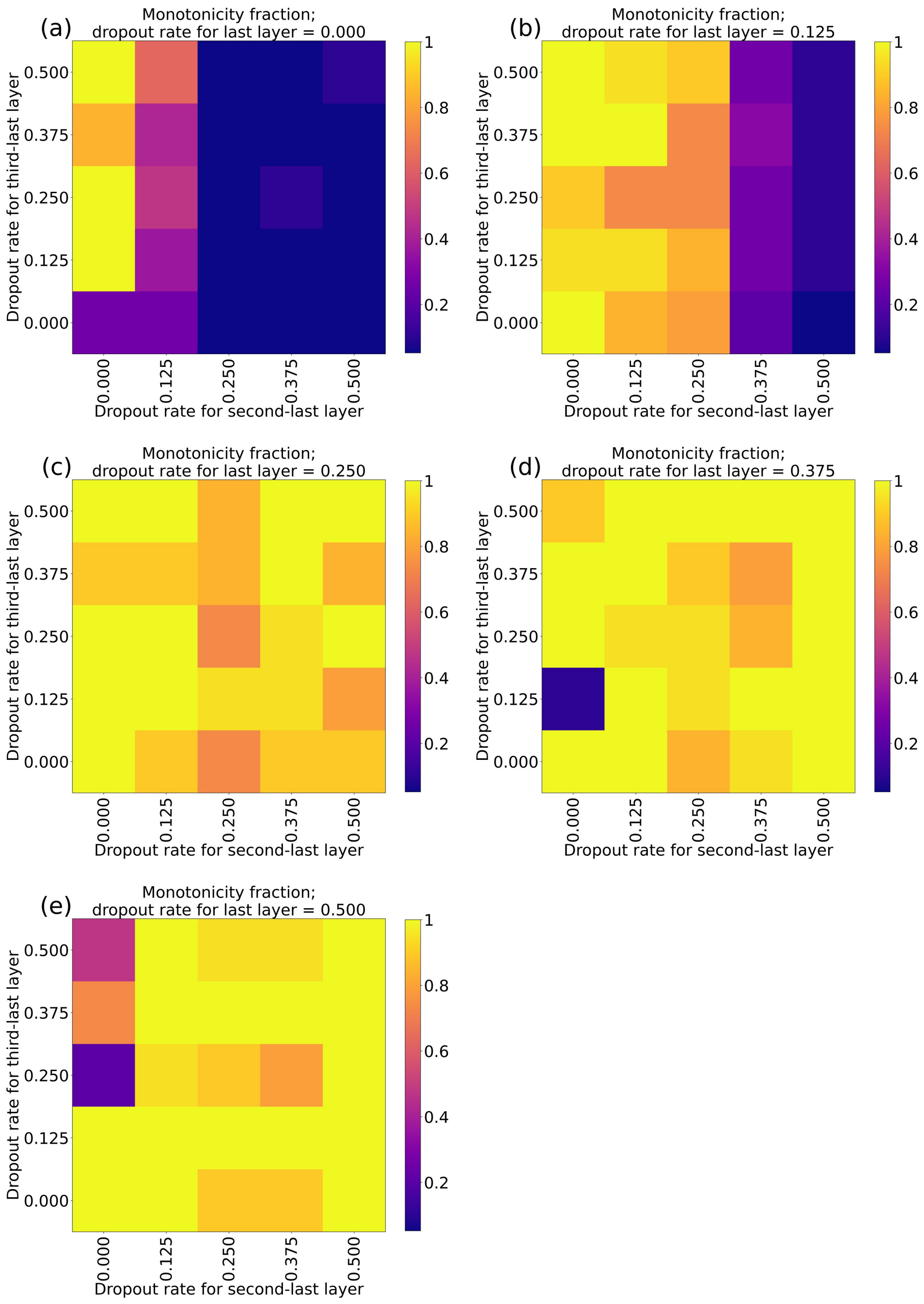


Figure S3: Monotonicity fraction, measured by running discard test on validation data, for models that use MC dropout to predict convection. Formatting is the same as in Figure S1.

b. Quantile regression (QR)

We attempt nine sets of quantile levels (Table S1) and ten weights (1, 2, 3, 4, 5, 6, 7, 8, 9, or 10) and use a grid search to attempt all $9 \times 10 = 90$ possible combinations. These weights (w in Eq. A2 of the appendix) affect the importance of deterministic vs. probabilistic predictions (*i.e.*, the mean prediction vs. quantile-based estimates) in the loss function. For all other hyperparameters, we use the settings documented in L21, including no dropout. To compute the model spread, defined as the standard deviation of the predictive distribution, we use Eq. 15 of Wan et al. (2014). As for the MC-dropout experiment, we use FSS to evaluate deterministic predictions, SSREL and MF to evaluate probabilistic predictions.

Table S1: Sets of quantile levels used in hyperparameter experiment. Every set also includes six special quantiles: 0.025 and 0.975 (used to compute the 95% confidence interval), 0.25 and 0.50 and 0.75 (used to compute the interquartile range and mean), and 0.99 (used to compute the psuedo-maximum, which is more robust than the actual maximum). These special quantiles are not listed in the table. “Increment” is the increment between successive quantile levels that are not in the special set; ”Total number” is the number of quantile levels, including those in the special set.

Increment	**Quantile levels**	**Total number**
0.01	0.01, 0.02, 0.03, ..., 0.99	101
0.02	0.01, 0.03, 0.05, ..., 0.99	53
0.03	0.01, 0.04, 0.07, ..., 0.97	38
0.04	0.01, 0.05, 0.09, ..., 0.97	30
0.05	0.01, 0.06, 0.11, ..., 0.96	26
0.06	0.01, 0.07, 0.13, ..., 0.97	22
0.07	0.01, 0.08, 0.15, ..., 0.99	19
0.08	0.01, 0.09, 0.17, ..., 0.97	18
0.09	0.01, 0.10, 0.19, ..., 0.91	17

Figs. S4-S6 show all three scores computed on the validation data, for all 90 models. All three figures are noisy, indicating little correlation between predictive quality and the hyperparameters. In our judgement, the best model is that with 17 quantile levels and a weight of 6 (w = 6 in Eq. A2 of the appendix), which achieves the 3rd-best FSS (0.264), the 7th-best SSREL (0.025), and a perfect MF (1.0).

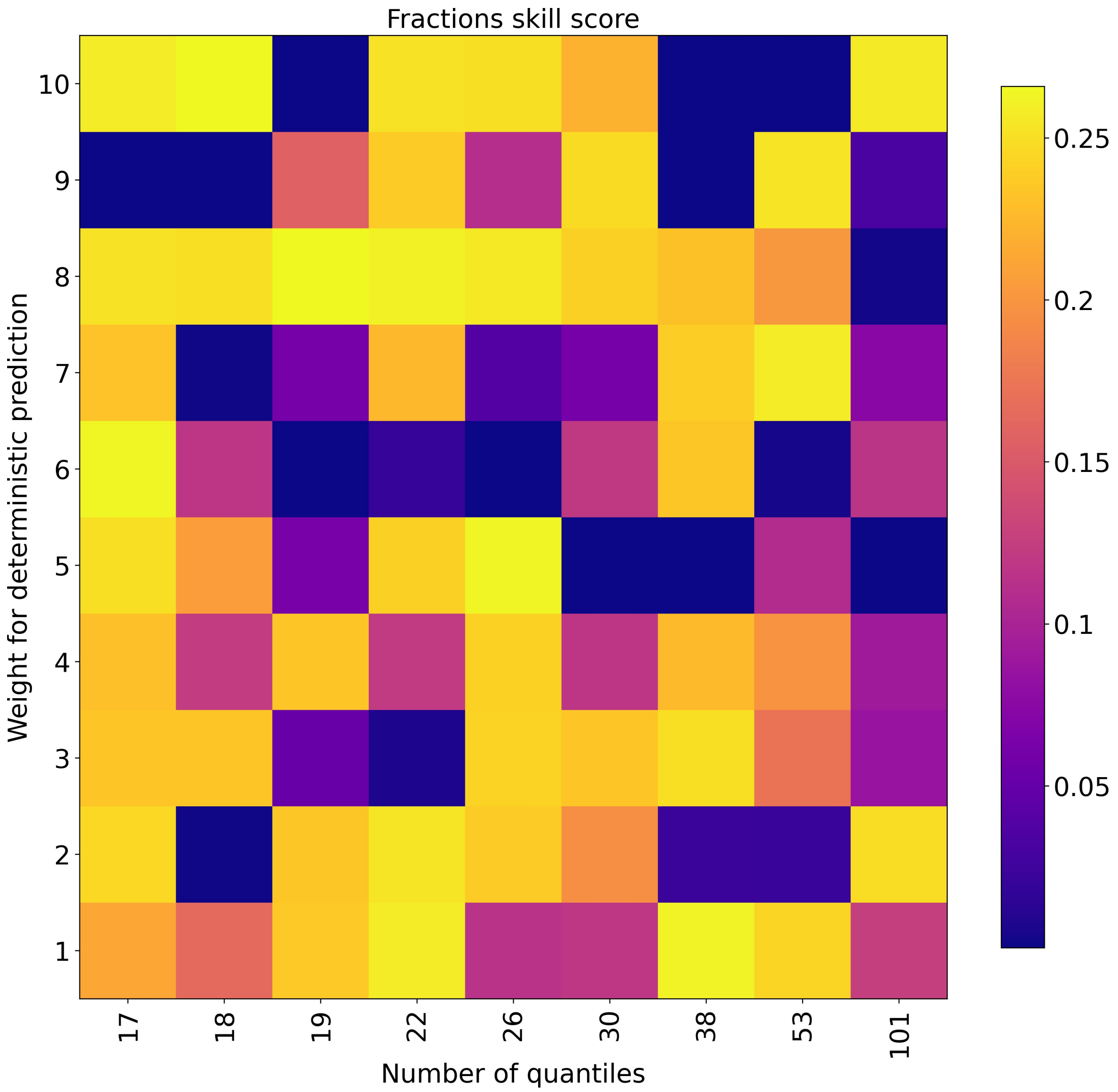


Figure S4: Pixelwise (1-by-1) fractions skill score (FSS), measured on validation data, when using QR to predict convection. The x-axis is the number of quantiles, and the y-axis is the weight used to emphasize the deterministic prediction in the loss function (w in Eq. A2 of the appendix).

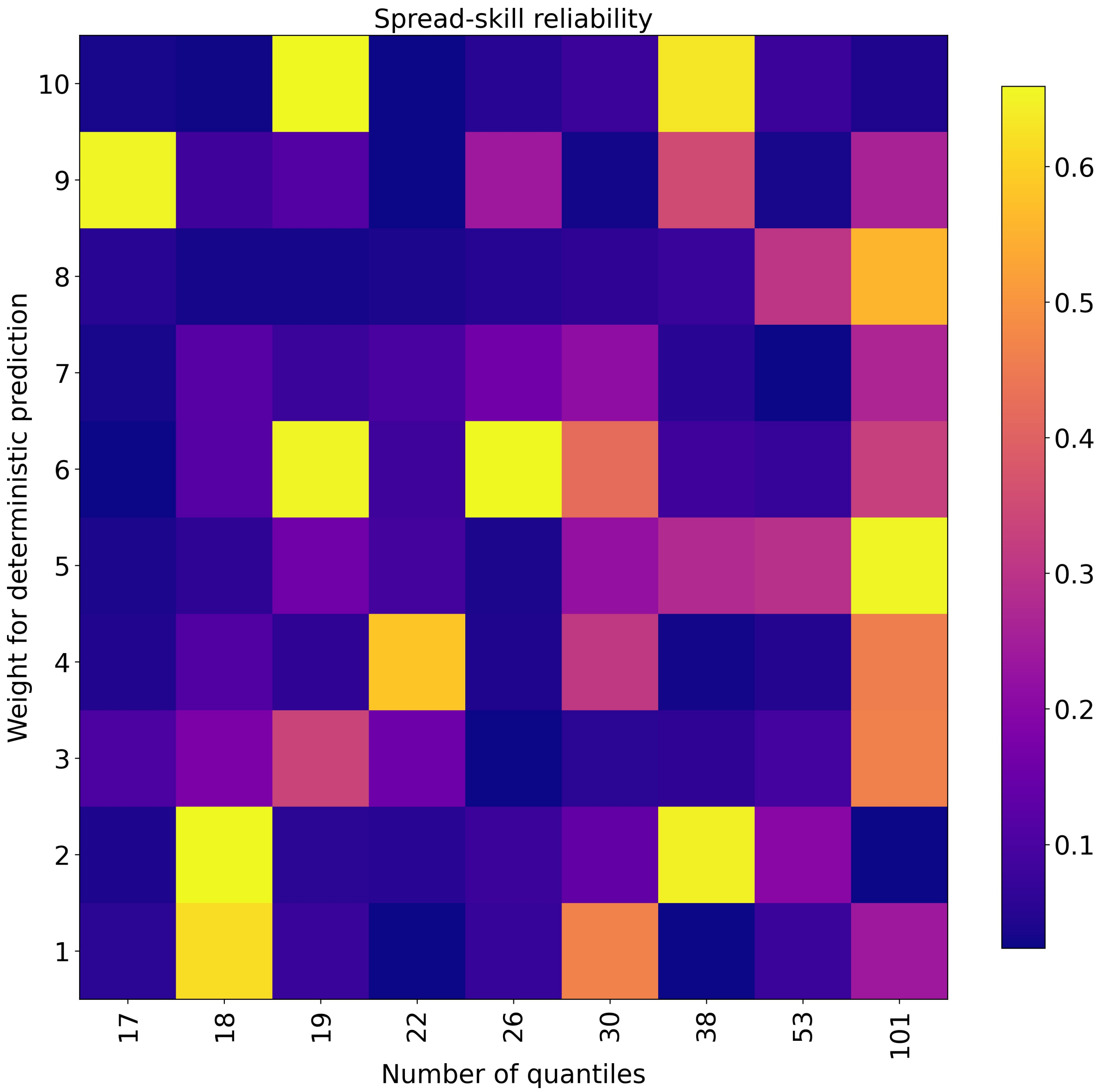


Figure S5: Spread-skill reliability (SSREL), measured on validation data, when using QR to predict convection. Formatting is the same as in Fig. S4.

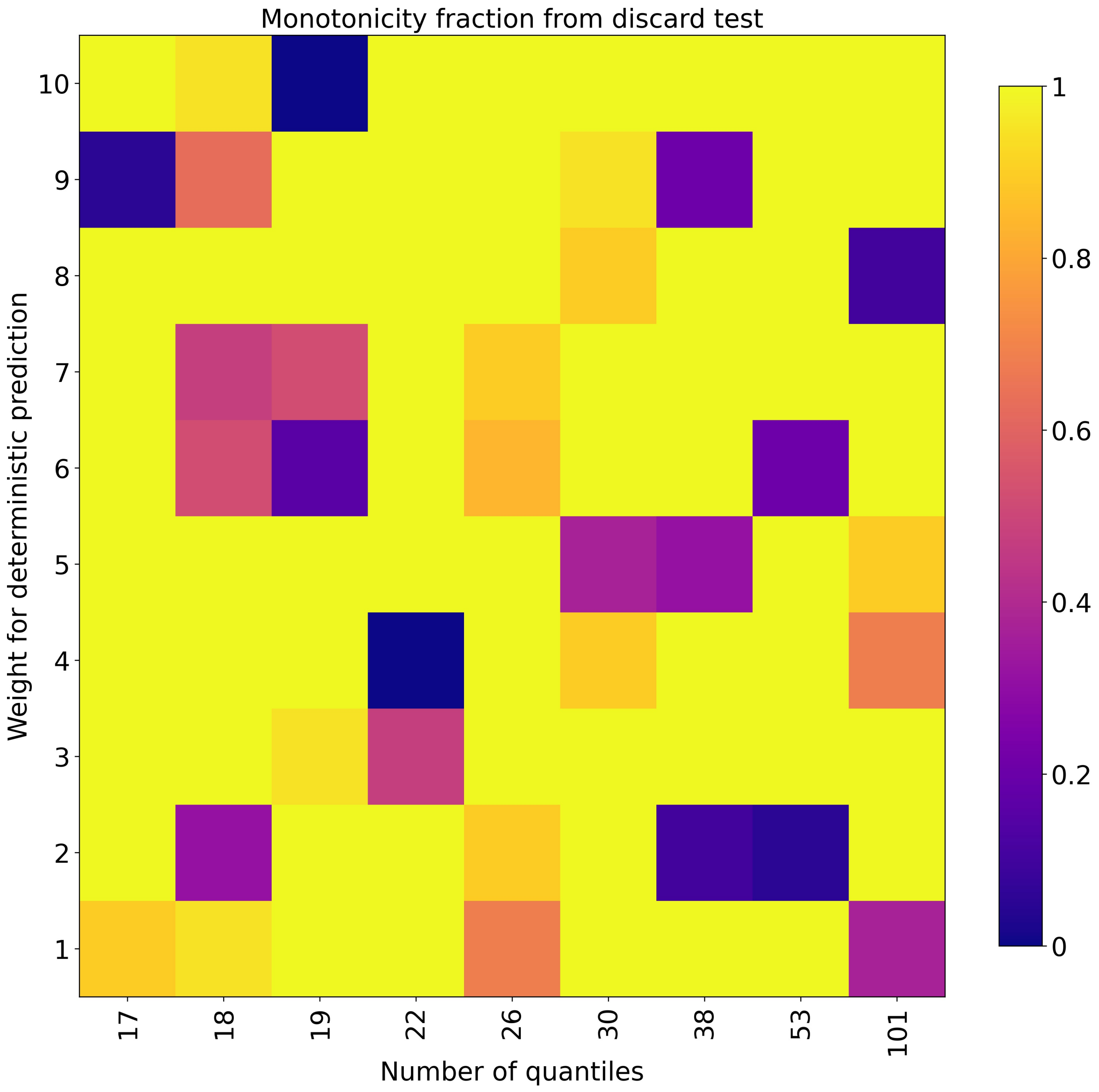


Figure S6: Monotonicity fraction, measured by running discard test on validation data, for models that use QR to predict convection. Formatting is the same as in Fig. S4.

2. Extra case study for classification task

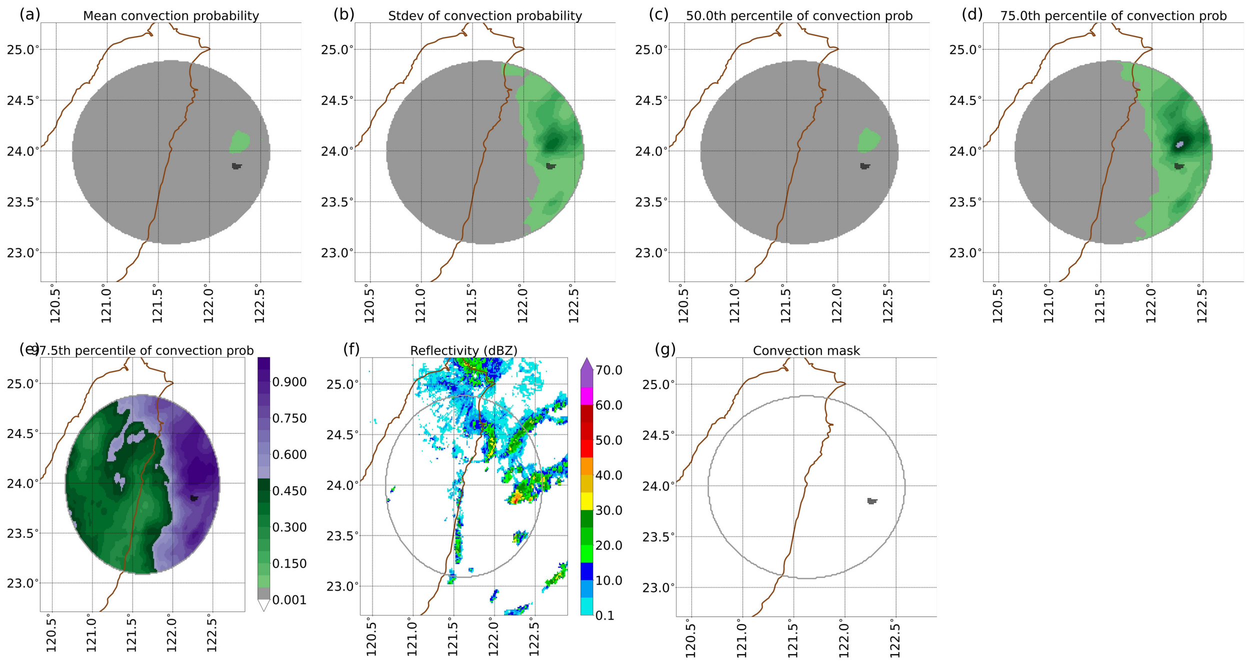


Figure S7: Winter case study for QR model. All data (predictions, radar reflectivity, and convection mask) are valid at 2230 UTC 25 Jan 2018. The predictions were made with 1-hour lead time, thus initialized at 2130 UTC. All formatting is discussed in the caption of Fig. 18 in the main body.

Fig. S7 shows a winter case for the QR model only[1]. There is only one thunderstorm in the domain and some scattered non-convective precipitation, mainly north and west of the one thunderstorm. This case illustrates two properties of the QR model. First, for the mean prediction (Fig. S7a) and lower percentiles of the predictive distribution (Fig. S7c-d), the model can produce very low probabilities over large areas. In other words, the model has sharpness at both low and high probabilities (the latter is shown in Fig. 19 in the main body, particularly panel e). The second property is a caveat to the first: at higher percentiles like the 97.5th (Fig. S7e), the QR model still produces very high probabilities, even in areas that are obviously[2] non-convective, due to the aforementioned overspread problem.

[1]The same case for the MC-dropout model is trivial, as the MC-dropout model produces almost no probabilities > 0.05, even at the 97.5th percentile.

[2]Based on visual analysis of the predictors (satellite brightness temperatures), not shown here.

References

Wan, X., W. Wang, J. Liu, and T. Tong, 2014: Estimating the sample mean and standard deviation from the sample size, median, range and/or interquartile range. *BMC Medical Research Methodology*, **14 (1)**, 1–13, URL https://doi.org/10.1186/1471-2288-14-135.