Machine-learned uncertainty quantification is not magic: Lessons learned from emulating radiative transfer with ML

This article was submitted to the Journal for Advances in Modeling Earth Systems *(JAMES) on Nov 8 2023.*

Ryan Lagerquist[1,2*], Imme Ebert-Uphoff[1,3], David D. Turner[2], and Jebb Q. Stewart[2]

[1]Cooperative Institute for Research in the Atmosphere (CIRA), Colorado State University, Fort Collins, Colorado
[2]National Oceanic and Atmospheric Administration (NOAA) Global Systems Laboratory (GSL), Boulder, Colorado
[3]Department of Electrical and Computer Engineering, Colorado State University, Fort Collins, Colorado

Key Points:

- Machine-learned uncertainty quantification (ML-UQ) is a new tool, and its limitations are not yet fully appreciated.
- Just like deterministic ML, ML-UQ often does not extrapolate well outside of the training domain.
- However, this problem can be somewhat mitigated by perturbing the training data along important basis vectors.

*325 Broadway, R/GSL6, Boulder, CO 80305

Corresponding author: Ryan Lagerquist, ralager@colostate.edu

Abstract

Machine-learned uncertainty quantification (ML-UQ) has become a hot topic in environmental science, especially for neural networks. Scientists foresee the use of ML-UQ to make better decisions and assess the trustworthiness of the ML model. However, because ML-UQ is a new tool, its limitations are not yet fully appreciated. For example, some types of uncertainty are fundamentally unresolvable, including uncertainty that arises from data being out of sample, *i.e.*, outside the distribution of the training data. While it is generally recognized that ML-based point predictions (predictions without UQ) do not extrapolate well out of sample, this awareness does not exist for ML-based uncertainty. When point predictions have a large error, instead of accounting for this error by producing a wider confidence interval, ML-UQ often fails just as spectacularly. We demonstrate this problem by training ML with five different UQ methods to predict shortwave radiative transfer. The ML-UQ models are trained with real data but then tasked with generalizing to perturbed data containing, *e.g.*, fictitious cloud and ozone layers. We show that ML-UQ completely fails on the perturbed data, which are far outside the training distribution. We also show that when the training data are lightly perturbed – so that each basis vector of perturbation has *a little* variation in the training data – ML-UQ can extrapolate along the basis vectors with some success, leading to much better (but still somewhat concerning) performance on the validation and testing data. Overall, we wish to discourage overreliance on ML-UQ, especially in operational environments.

Plain-language summary

Machine-learned uncertainty quantification (ML-UQ) – *i.e.*, ML models that return both a point prediction and an estimate of their own uncertainty – is a hot topic in environmental science. Recent developments in ML-UQ have generated much excitement, but this excitement should be tempered by an awareness of its limitations. For example, just like basic ML (with only point predictions) extrapolates poorly outside the distribution of its training data, so do uncertainty estimates from ML-UQ. This can lead to catastrophic errors, *i.e.*, very wrong predictions made with high confidence (low uncertainty). We demonstrate this problem across a range of ML-UQ methods and address a way to alleviate the problem.

1 Introduction

1.1 Machine-learned uncertainty in environmental science

For as long as machine learning (ML) has been used in environmental science (ES), both developers and users have been interested in how *uncertain* the predictions are. This uncertainty quantification (UQ) is especially important for high-impact applications, such as severe weather, where an incorrect prediction can cost property and human lives. The computer-science literature has recently made breakthroughs in ML models that quantify their own uncertainty (ML-UQ), which could be a game-changer for high-impact ML applications in ES. The next step is for the ES community to familiarize itself with these new ML-UQ tools and modify them to best suit the unique needs of ES applications. This work is already in progress (Rasp et al., 2018; Wimmers et al., 2019; Scheuerer et al., 2020; Baran & Baran, 2021; Bihlo, 2021; Barnes et al., 2021; Clare et al., 2021; Ghazvinian et al., 2021; Orescanin et al., 2021; Scher & Messori, 2021; Veldkamp et al., 2021; Chapman et al., 2022; Garg et al., 2022; Klotz et al., 2022; Ortiz et al., 2022; Schulz & Lerch, 2022). Specifically, the ES community is asking the following questions:

1. Which ML-UQ methods are reasonably easy to learn and implement?
2. Which methods are available for classification (predicting a category) vs. regression (predicting a continuous real number)? The computer-science community of-

ten develops methods for classification tasks, whereas many ES problems are regression tasks.
3. What is the best ML-UQ method for a given application? Just like the quality of point predictions (*i.e.*, single predictions without UQ, often called "deterministic") can be evaluated with objective tools, so can the quality of uncertainty estimates. See Haynes et al. (2023, henceforth H23) for an overview of UQ evaluation.

We are particularly interested in the last question. Specifically, we take one step further back and ask:

1. Are there fundamentally *unresolvable* types of uncertainty (*i.e.*, that cannot be captured with any UQ method)?
2. If so, what real-world scenarios create unresolvable uncertainty? What are the implications for using ML-UQ in operations? For example, how should the uncertainty estimates be interpreted, knowing that they might completely miss some types of uncertainty?

We are interested in these questions because we foresee a danger of scientists relying too heavily on ML-UQ.

1.2 ML-UQ is not magic

To understand why we are concerned, let us briefly recap the state of ML in ES. ML has shown great promise in terms of improved accuracy and faster execution, relative to process-based models. Although these advantages have been demonstrated for many ES applications, users, such as operational weather-forecasters, have been slow to accept ML into operations (Gil et al., 2019; Reichstein et al., 2019). The primary reason is that ML is not guaranteed to generalize well to out-of-sample data (Buiten, 2019) – *e.g.*, locations, seasons, or physical regimes that were not included in the training data. In contrast, process-based models, which employ known laws of physics, typically generalize much better out-of-sample. Also, where process-based models make an approximation, users generally understand the potential impacts – *e.g.*, situations where the model is thereby inappropriate. This understanding is much harder to build for ML models.

One hope is that recently developed ML-UQ methods can help indicate situations where an ML model is inappropriate, *i.e.*, where it will produce unacceptable errors. However, this hope rests on the assumption that the model's estimates of its own uncertainty are highly correlated with its error – *i.e.*, that the model "knows when it is wrong". It is our subjective experience that scientists do not question an ML model's uncertainty estimates as much as they question its predictions. In particular, scientists do not consider the possibility of *catastrophic errors*: extremely wrong predictions made with high confidence (low uncertainty). The concept of catastrophic errors, especially arising due to unresolvable uncertainty, is at the core of this manuscript.

1.3 A few examples of unresolvable uncertainty

An ML-based UQ method (*e.g.*, Bayesian neural network) must ground its uncertainty estimates in the training data, just like the base ML model (*e.g.*, neural network) must ground its predictions in the training data. No other information is provided to the model. Thus, if a physical relationship exists in the real world but is not represented in the training data, it will not be learned by the base model or ML-UQ method. From this insight, we construct three scenarios that *any* ML-UQ method would struggle with.

Scenario 1: Missing variable. The target variable y depends strongly on a variable not included in the predictors. This scenario is common, as some variables cannot be measured and a limited number of variables can be included in an ML model. Specifically, consider an example where y is a function of two variables, x_{known} and x_{unknown}, but only x_{known} is included in the predictors. Let the model be $\hat{f}$ with a probabilistic output vector $\vec{y}_{\text{pred}}$, which represents the full predicted distribution. (For example, if $\hat{f}$ is an ensemble model, each element of $\vec{y}_{\text{pred}}$ is one member of the ensemble; if $\hat{f}$ is a parametric model assuming the normal distribution, the two elements of $\vec{y}_{\text{pred}}$ are mean and variance; ...; etc.)

$$\begin{cases} y_{\text{true}}(x_{\text{known}}, x_{\text{unknown}}) = x_{\text{known}} \cdot x_{\text{unknown}}; \\ \vec{y}_{\text{pred}}(x_{\text{known}}) = \hat{f}(x_{\text{known}}). \end{cases} \tag{1}$$

Since y_{true} depends strongly on x_{unknown} but the model does not have access to x_{unknown}, the distribution $\vec{y}_{\text{pred}}$ – including any point prediction (*e.g.*, the mean) and any measure of uncertainty (*e.g.*, the variance) – will lack skill. In other words, the model's point predictions will be poor, and the tool designed to alert us when point predictions are poor – namely UQ – will fail as well. Although this example is extreme, unresolvable uncertainty can arise in other ways. The point of this example is to illustrate that *any* UQ method will fail to alert us to the model's poor predictions.

Scenario 2: Variable constant in training data. y depends strongly on a variable x_c that, although it is included in the predictors, takes a constant value over all the training data. In general, though, x_c is not constant. Specifically, consider an example with one other predictor, x:

$$\begin{cases} y_{\text{true}}(x, x_c) = x \cdot x_c; \\ \vec{y}_{\text{pred}}(x, x_c) = \hat{f}(x, x_c) = \hat{f}(x). \end{cases} \tag{2}$$

Although both x and x_c are provided to the model $\hat{f}$, it cannot learn anything from a variable that does not actually *vary* in the training data. Replacing x with x_{known} and x_c with x_{unknown}, Equation 2 becomes Equation 1, so uncertainty arising from x_c is unresolvable. For a more intuitive example, consider a climate model trained to predict global-annual-averaged temperature (GAAT), with one of the predictors being CO_2 concentration (q). All training samples contain the year-2000 value, $q = 370$ ppm; but the model is then applied to year-2100 data, with $q = 600$ ppm. The year-2100 data are *out of sample* with respect to q, leading to unresolvable uncertainty and catastrophic errors. Specifically, the model will severely underpredict GAAT with high confidence.

Scenario 3: Missing basis vector in training data. y depends strongly on variations along a basis vector $\hat{b}$ of the predictor space, but the training data contain no variation along this direction. Scenario 3 is a more general example of scenario 2, where $\hat{b} = \hat{x}_c$. Scenario 2 is unlikely because it is easy to spot (*e.g.*, by plotting a histogram of every predictor variable), whereas scenario 3 is hard to spot, because it is hard to know all the important basis vectors in a dataset, especially for high-dimensional data. As our experiments in Sections 5 and 6 show, if an important basis vector (*e.g.*, thickness of the ozone layer) is not well sampled in the training data, this can lead to catastrophic out-of-sample errors. (We use the term *basis vector* loosely; in the strict definition all basis vectors of a space are orthogonal, which is not necessarily true in our data. The term "latent variable" or "latent-space vector" would be more accurate (Van et al., 2020), since latent spaces do not imply orthogonality, but we feel that "basis vector" is more familiar to an ES audience.)

1.4 Our sample application: Shortwave radiative transfer

Simulating radiative transfer (RT) – *i.e.*, heating of the atmosphere due to the scattering and absorption of radiation by particles such as hydrometeors, water vapour, aerosols,

and trace gases – is a key part of numerical weather prediction (NWP). However, existing RT models are computationally expensive, which slows down NWP. In previous work (Lagerquist et al., 2023, henceforth L23) we demonstrated that one of these models, namely the Rapid Radiative Transfer Model (RRTM; Iacono et al., 2008), can be emulated accurately and quickly with neural networks. The current study builds on L23 and focuses entirely on UQ, which is not included in L23 or the RRTM. We focus on shortwave radiation (wavelengths of 0.2-12.2 μm), which is largely of solar origin.

Note that the goal of this paper is *not* to generate new insights for the application of emulating RT. Rather, we use this application because it is an ideal setup to experiment with the scenarios of unresolvable uncertainty discussed above. Because we are using ML to emulate another model (the RRTM), we can freely modify the inputs (predictors) and use the RRTM to compute the corresponding correct outputs (targets). We use this setup to explore the following questions:

1. Can we observe the theoretical scenarios from Section 1.3 in practice? *i.e.*, Can we cause ML-UQ to fail catastrophically for such scenarios?
2. How drastic are the failures in practice? Do some UQ methods fare better than others? Are there simple ways to address the failures?

1.5 Organization of this manuscript

First, we create out-of-sample data that should confound *any* ML-UQ method. Specifically, we perturb the validation and testing data along several basis vectors (*e.g.*, thickness of the ozone layer) that have very little variability in the training data. Second, we train models with five different ML-UQ methods and apply them to the perturbed validation and testing data, verifying that all five methods produce catastrophic errors. Third, we explore whether we can reduce these catastrophic errors by perturbing the training data *just a little* along each basis vector.

2 Input data

This section is a brief overview of the predictor and target variables, referring to L23 for details. The RRTM and ML-based emulators have the same target variables and mostly the same predictor variables; the emulators have two extra predictors, for reasons discussed in Section 2a of L23. For the target variables, values produced by the RRTM are considered ground truth, or "labels" in ML terminology.

2.1 Predictor variables

We use 26 predictor variables, summarized in Table 1. Most of these variables are available in output files from version 16 of the Global Forecast System model (GFSv16; see 2021 update at `https://www.emc.ncep.noaa.gov/emc/pages/numerical_forecast_systems/gfs/documentation.php`), but a few are not. For these synthetic variables, we create fictitious data, following Section 2b of L23. For the GFSv16 variables, we extract forecast profiles at locations around the globe from 0000 UTC model runs on dates from Sep 1 2018 to Dec 23 2020. Thus, our dataset is global in terms of both geographic location and seasonality – *i.e.*, covers all times of year at all locations. For more details on the GFSv16 variables, see Section 2a of L23.

Table 1: Description of predictor variables. "Scalar?" indicates whether the variable is scalar, versus a full profile. "Synthetic?" indicates whether the values are synthesized from fake data, versus taken from GFSv16 output. "ML only?" indicates whether the variable is used only in the ML-based emulators, versus both ML and RRTM. "AGL" = above ground level. Downward LWP at height z is LWC integrated from the top of the profile down to z, and upward LWP at height z is LWC integrated from the bottom of the profile up to z. Downward IWP, upward IWP, downward WVP, and upward WVP have analogous definitions.

Variable	**Units**	**Scalar?**	**Synthetic?**	**ML only?**
Temperature	K			
Pressure	Pa			
Specific humidity	kg kg^{-1}			
Relative humidity	—			
Liquid-water content (LWC)	kg m^{-3}			
Ice-water content (LWC)	kg m^{-3}			
Downward liquid-water path (LWP)	kg m^{-2}			
Downward ice-water path (IWP)	kg m^{-2}			
Downward water-vapour path (WVP)	kg m^{-2}			
Upward LWP	kg m^{-2}			
Upward IWP	kg m^{-2}			
Upward WVP	kg m^{-2}			
O_3 mixing ratio	kg kg^{-1}			
Height	m AGL			
Solar zenith angle	°	✓		
Surface albedo	—	✓		
Height thickness	m			✓
Pressure thickness	Pa			✓
Aerosol single-scattering albedo	—	✓	✓	
Aerosol asymmetry parameter	—	✓	✓	
Aerosol extinction coefficient	m^{-1}		✓	
Liquid effective radius	m		✓	
Ice effective radius	m		✓	
N_2O concentration	ppmv		✓	
CH_4 concentration	ppmv		✓	
CO_2 concentration	ppmv		✓	

2.2 Target variables

The RRTM performs 1-dimensional RT, assuming that RT occurs only in the vertical. Thus, both the RRTM and emulators are applied to each profile separately. The target variables are those required by an NWP model from its RT parameterization: a full profile of heating rates (HR), surface downwelling flux ($F_{\text{down}}^{\text{sfc}}$), top-of-atmosphere upwelling flux ($F_{\text{up}}^{\text{TOA}}$), and net flux ($F_{\text{net}}$). See Figure 1 for an example.

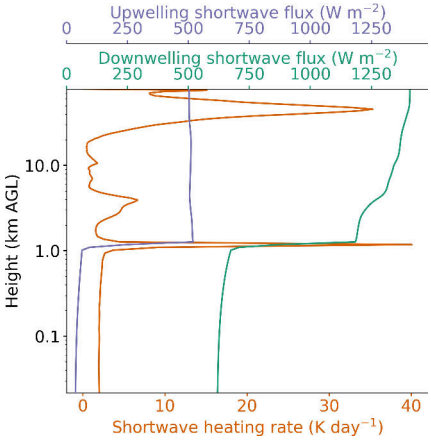


Figure 1: RRTM outputs for one data sample. We emulate the full profile of heating rates, $F_{\text{down}}^{\text{sfc}}$ (the bottom value in the green curve), $F_{\text{up}}^{\text{TOA}}$ (the top value in the purple curve), and F_{net} (the difference between the last two values).

2.3 Preparing the data for ML

Our data preparation includes three steps. First, we split the data into three temporally independent partitions: training, validation, and testing (Table 2). We use the training data to optimize parameters (weights and biases) for each ML model, the validation data to select the best ML model (*e.g.*, best UQ method), and the testing data for a final assessment of the selected model. Second, we perturb the data in each partition to a different extent: the training data *not at all* (for the first experiment) or *lightly* (for the second experiment), the validation data *moderately*, and the testing data *heavily*. In other words, the ML models are trained with clean or lightly perturbed data, selected based on moderately perturbed data, and then tasked with generalizing to heavily perturbed data. Third, we normalize each predictor variable from physical units to z-scores, following Section 3b of Lagerquist et al. (2021).

Table 2: Partitioning of data into temporally independent subsets. "Sample size" = number of profiles. Also, "Number of days" ≠ length of "Time period," because some days are missing from the archive.

Data subset	**Time period**		**Number of days**	**Sample size**
Training	Sep 1 2018 – Dec 21 2019		237	873 086
Validation	Jan 2-15 2020, Mar 24 – Apr 6 2020, Jun 16-29 2020, Sep 6-19 2020, Nov 30 – Dec 13 2020	Feb 12-25 2020, May 5-18 2020, Jul 27 – Aug 9 2020, Oct 19 – Nov 2 2020,	126	479 806
Testing	Jan 18-31 2020, Apr 9-22 2020, Jul 2-15 2020, Sep 22 – Oct 7 2020, Dec 16-23 2020	Feb 28 – Mar 12 2020, May 22 – Jun 4 2020, Aug 12-25 2020, Nov 5-18 2020,	120	474 726

2.4 Perturbing to create out-of-sample data

We create out-of-sample data by perturbing five atmospheric properties represented in the predictor variables. The five properties are near-surface temperature, near-surface humidity, liquid cloud, ice cloud, and ozone. Loosely, each property may be seen as corresponding to one or more basis vectors of the predictor space. Some of our perturbations – *e.g.*, increasing near-surface temperature and humidity – mimic impacts that are expected from climate change, a real process that creates out-of-sample data. Some researchers have developed methods to make ML more robust to climate change (Beucler et al., 2021), albeit with a focus on point predictions rather than uncertainty estimates. However, some of our perturbations – *e.g.*, those involving the ozone layer – are unlike anything seen in the Earth's atmosphere or expected with climate change. Supplemental Figures S5-S9 show the distribution of each variable before and after perturbation; here it is evident, for example, that the changes to ozone are much more extreme than the changes to temperature and humidity. These *extreme* perturbations allow us to observe the behaviour of the UQ methods when tasked with generalizing to *extremely* out-of-sample data. In other words, the more extreme perturbations allow us to stress-test the UQ methods in a way that more realistic data would not.

The target values – *i.e.*, heating rates and fluxes – must change in response to the new predictors. To obtain the new target values ($\vec{y'}$) for a given profile, we simply feed the new predictors ($\vec{x'}$) to the RRTM.

Two details remain to be specified: [1] Which atmospheric properties are perturbed for which profiles? [2] What are the specific perturbation methods? For each profile P and each property χ, there is a 50% chance that χ will be perturbed in P, based on drawing a random integer from $\{0, 1\}$. For a given profile P, if all five random numbers evaluate to 0, one of the five is randomly changed to 1, so that at least one property is perturbed for every profile. The subsections below explain the specific perturbation method for each atmospheric property.

Near-surface temperature

Our motivation for this procedure is to mimic the lower-tropospheric warming expected with climate change. The procedure has two parameters: maximum depth of the warm layer ($D_{\max}$) and maximum surface-temperature increase ($\Delta T_{\text{sfc}}^{\max}$). For the lightly perturbed training data, we set $D_{\max} = 1.25$ km and $\Delta T_{\text{sfc}}^{\max} = 2$ K; for the moderately perturbed validation data, $D_{\max} = 2.5$ km and $\Delta T_{\text{sfc}}^{\max} = 4$ K; for the heavily perturbed testing data, $D_{\max} = 5$ km and $\Delta T_{\text{sfc}}^{\max} = 8$ K. The procedure is shown schematically in Figure 2. After the numbered procedure below, we recompute relative humidity, based on the new temperature and untouched specific humidity.

1. Determine the depth of the warm layer by sampling from a uniform distribution over $[0, D_{\max}]$. Symbolically, $D \in \mathcal{U}[0, D_{\max}]$.
2. Sample to determine the surface-temperature increase: $\Delta T_{\text{sfc}} \in \mathcal{U}[0, \Delta T_{\text{sfc}}^{\max}]$.
3. At each height in the warm layer, scale the temperature increase linearly from ΔT_{sfc} at the surface to 0 at height D above the surface. See Figures 2a-c.
4. If step 3 led to any temperature above 60 °C, reduce to 60 °C. See Figure 2d.

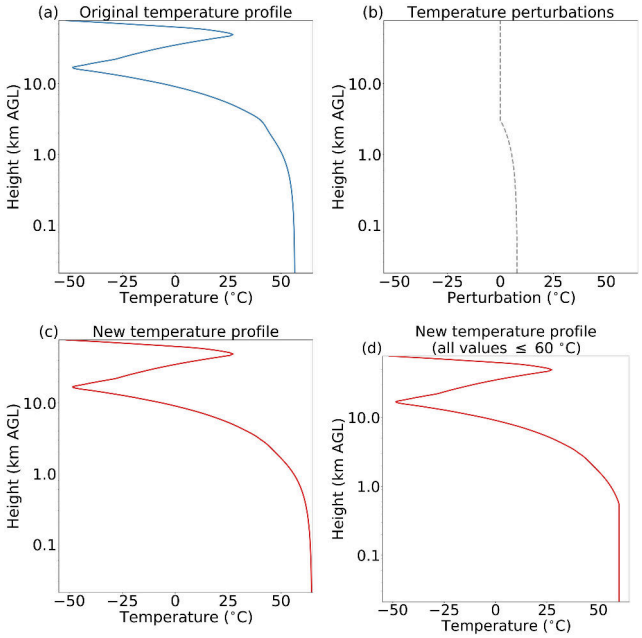


Figure 2: Procedure for perturbing near-surface temperature. Panel c = a + b. In this example, the warm-layer depth D is 3 km and the surface-temperature increase ΔT_{sfc} is 8 K.

Near-surface humidity

Our motivation is to mimic the lower-tropospheric moistening expected with climate change. We first generate a disturbance for the relative humidity (RH) profile, then recompute the other moisture variable (specific humidity) from the new RH. See Supplemental Section 1 for details.

Liquid cloud

Our motivation is to create more complex cloud arrangements, as well as denser and deeper clouds, than seen in the real atmosphere. We completely replace the liquid-water content (LWC) profile, generating a number of cloud layers from 0 up to $N_{\max}$. $N_{\max}$ varies from 2 for the lightly perturbed training data to 5 for the heavily perturbed testing data. See Supplemental Section 1 for details.

Ice cloud

The motivation for perturbing ice cloud is the same as for perturbing liquid cloud; the two procedures are nearly identical. See Supplemental Section 1 for details.

Ozone

Our motivation is to create more complex ozone layers – over a wider range of locations, depths, and mixing ratios – than seen in the real atmosphere. We completely replace the ozone mixing ratio (w) profile, generating an ozone layer with a random location, depth, and structure. See Supplemental Section 1 for details.

3 Methods

3.1 The base model: U-net++

The field of deep learning has produced many specialized neural network (NN) architectures for handling spatial data. We have chosen the U-net++ architecture, which L23 found to be the best for shortwave RT. The U-net++ is a slight generalization of the U-net (Ronneberger et al., 2015), which is designed for image-to-image translation, *i.e.*, to output predictions on the same spatial grid as the predictors. The U-net contains four key components: convolutional layers, pooling (downsampling) layers, upsampling layers, and skip connections. Convolutional layers use learned image filters to detect spatial and multivariate features in the predictor data, producing abstract representations of the predictor data, called "feature maps". Pooling and upsampling layers scale feature maps to coarser and finer spatial resolutions, respectively, allowing convolutional layers to detect features at different scales. Skip connections carry high-resolution feature maps directly across the network, bypassing the series of downsampling and upsampling layers, which is a lossy operation that degrades high-resolution information. The U-net++ (Zhou et al., 2019) is a U-net with more skip connections, which more effectively preserve small-scale features, such as cloud boundaries, that are important for shortwave RT. Our specific U-net++ setup for point prediction is shown in Figure 3. Our main learning task is to translate a 127-by-26 image of predictor variables into a 127-by-1 image of heating rates. (There are 127 heights in the GFS grid and 26 predictor variables; see Table 1. We duplicate the 4 scalar variables over all 127 heights, so that they fit into the matrix.) There is also a second learning task: to predict the three flux variables ($F_{\text{down}}^{\text{sfc}}$, $F_{\text{up}}^{\text{TOA}}$, and F_{net}), which are scalars rather than images. For this we attach fully connected layers – which are used in traditional (non-convolutional) NNs (Chapter 6 of Goodfellow et al., 2016) and still a popular choice for scalar data – to the U-net++.

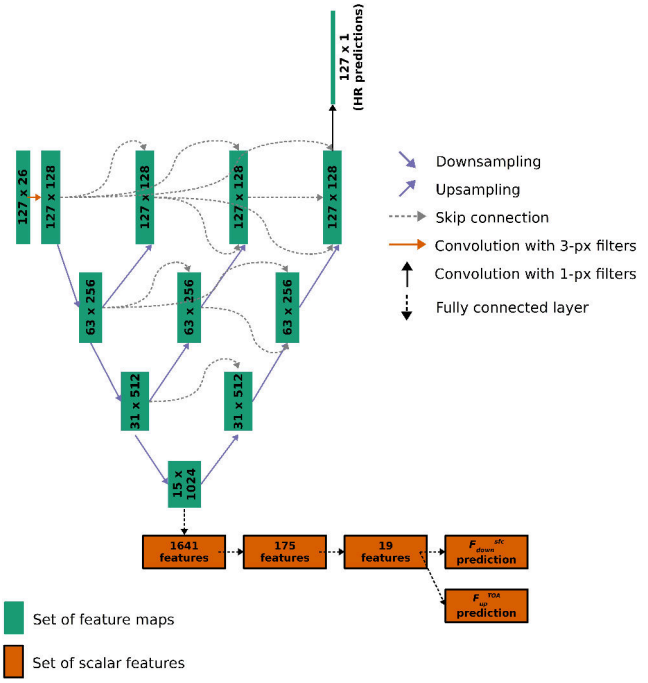


Figure 3: [adapted from Figure 3a of L23] Our specific U-net++ setup for point prediction. For each set of feature maps (green box), the label is number_of_heights × number_of_channels. In the remaining discussion, let K be the number of convolutional layers per block. We call this hyperparameter "width" in L23; the chosen value in this study, based on L23, is $K = 1$. Each orange "convolution" arrow represents K convolutional layers with 3-pixel filters; each "downsampling" arrow represents K convolutional layers with 3-pixel filters, followed by maximum-pooling with a 2-pixel window; each "upsampling" arrow represents upsampling with a 2-pixel window, followed by a convolutional layer with 3-pixel filters; each "skip connection" arrow includes K convolutional layers with 3-pixel filters; each black "convolution" arrow represents one convolutional layer with 1-pixel filters; and finally, each "fully connected layer" arrow represents one fully connected layer.

3.2 The ML-UQ methods

The total uncertainty in an ML model is the sum of two components: aleatory and epistemic. The Appendix provides definitions of these terms – which, interestingly, differ across disciplines – and shows that the examples of unresolvable uncertainty from Section 1.3 can show up in both components. Thus, our analysis must include UQ methods that can capture both the aleatory and epistemic components of uncertainty. Specifically, we use the three UQ methods discussed in the subsections. The first method (CRPS-LF) was found by H23 to perform well, but it can capture only aleatory uncertainty. Thus, we also use the multi-model ensemble (MME) and Bayesian neural networks (BNN). On their own MME and BNN can capture only epistemic uncertainty, but either method can be combined with CRPS-LF to capture both types of uncertainty.

How to read this section: Our purpose for testing multiple UQ methods is to show that our results generalize across UQ methods. The interested reader may continue with this section and see H23 for even more details; readers less interested in the inner workings of UQ methods may skip ahead to Section 4.

3.2.1 CRPS-LF

This approach involves training a NN with the continuous ranked probability score (CRPS) as the loss function (LF). The CRPS-LF approach can be used with both parametric prediction and ensemble prediction. In ensemble prediction, the NN approximates y_{pred} by generating an ensemble, and its loss function is the ensemble formulation of the CRPS:

$$\text{CRPS} = \frac{1}{N}\sum_{i=1}^{N}|y_{\text{true}} - y_{\text{pred}}^{i}| - \frac{1}{2}\frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}|y_{\text{pred}}^{i} - y_{\text{pred}}^{j}|, \tag{3}$$

where N is the ensemble size; y_{true} is the correct value; and y_{pred}^{k} is the k^{th} prediction in the ensemble. The first term is the mean absolute error (MAE), and the second is the mean absolute pairwise difference (MAPD) between ensemble members, a measure of spread. The CRPS ranges from $[0, \infty)$; the optimal value is 0.

The CRPS is an uncertainty-oriented generalization of the MAE, which is a standard loss function for point prediction. However, for point prediction we use a custom loss function to emphasize large heating rates (Section 3d of L23), which the NN predicts poorly when trained with standard loss functions. Specifically, we use the following loss function for point prediction:

$$\mathcal{L} = \frac{1}{H}\sum_{h=1}^{H}\max\left\{|r_h|, |\hat{r}_h|\right\}(r_h - \hat{r}_h)^2 + \frac{1}{L}\sum_{l=1}^{L}(F_l - \hat{F}_l)^2, \tag{4}$$

where $H = 127$ is the number of heights; r_h is the actual heating rate at the h^{th} height; $\hat{r}_h$ is the corresponding prediction; $L = 3$ is the number of flux variables; F_l is the actual value of the l^{th} flux variable; and $\hat{F}_l$ is the corresponding prediction. The second term is the standard MSE for flux variables, but the first term is a weighted MSE for heating rates, the weight being $\max\left\{|r_h|, |\hat{r}_h|\right\}$. We call this term the dual-weighted MSE (DWMSE).

To generalize the above loss function for UQ, we hybridize Equations 3 and 4, yielding the dual-weighted CRPS (DWCRPS):

$$\text{DWCRPS} = \frac{1}{H}\frac{1}{N}\sum_{h=1}^{H}\sum_{i=1}^{N}\max\left\{|r_h|, |\hat{r}_{hi}|\right\}|r_h - \hat{r}_{hi}| - \frac{1}{2}\frac{1}{H}\frac{1}{N^2}\sum_{h=1}^{H}\sum_{i=1}^{N}\sum_{j=1}^{N}\max\left\{|\hat{r}_{hi}|, |\hat{r}_{hj}|\right\}|\hat{r}_{hi} - \hat{r}_{hj}|. \tag{5}$$

H, N, and r_h are as defined previously; $\hat{r}_{hk}$ is the k^{th} predicted heating rate at the h^{th} height; $\max\left\{|r_h|, |\hat{r}_{hi}|\right\}$ is the maximum absolute value of the actual and i^{th} predicted heating rate at the h^{th} height; and $\max\left\{|\hat{r}_{hi}|, |\hat{r}_{hj}|\right\}$ is the maximum absolute value of the i^{th} and j^{th} predicted heating rates at the h^{th} height. Both max terms are weights that emphasize large heating rates.

The DWCRPS is used only for heating rates; the standard CRPS is used for fluxes, since the distribution of fluxes is less skewed (Figure 5 of Lagerquist et al., 2021) and therefore does not necessitate a custom loss function to ensure good prediction of extreme values. Thus, the total loss function we use for the CRPS-LF approach is:

$$\mathcal{L} = \text{DWCRPS} + \frac{1}{L}\frac{1}{N}\sum_{l=1}^{L}\sum_{i=1}^{N}|F_l - \hat{F}_{li}| - \frac{1}{2}\frac{1}{L}\frac{1}{N}\sum_{l=1}^{L}\sum_{i=1}^{N}\sum_{j=1}^{N}|\hat{F}_{li} - \hat{F}_{lj}|. \tag{6}$$

The second and third terms, collectively, are the CRPS for flux variables.

To make the CRPS-LF approach work, in addition to changing the loss function, one must change the NN architecture to output N estimates per target variable (Figure 4).

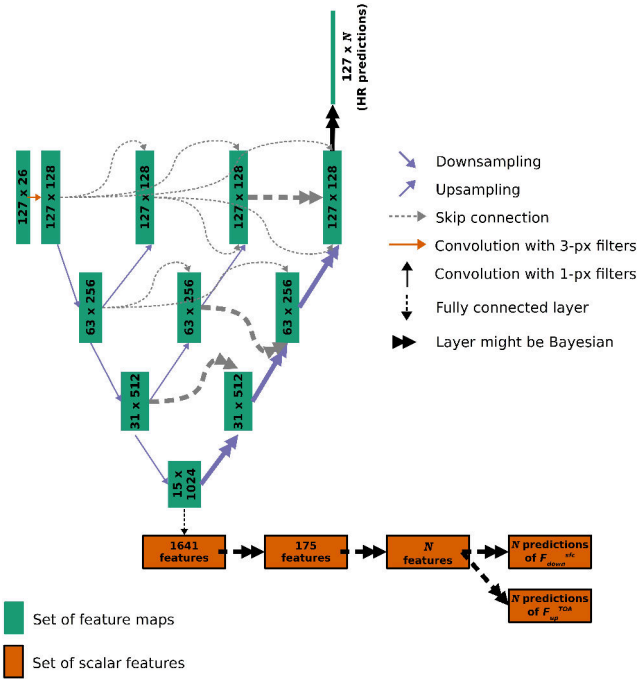


Figure 4: [adapted from Figure 3a of L23] Our specific U-net++ setup for UQ. If using the CRPS-LF approach, N (the ensemble size) > 1 and the loss function is Equation 6; otherwise, $N = 1$ and the loss function is Equation 4. If using the BNN approach, one or more of the double arrows contain Bayesian layers. The arrows marked "fully connected layer" and "convolution with 1-px filters" each represent a single layer; the corresponding layer may or may not be Bayesian. Meanwhile, recall from the caption of Figure 3 that the arrows marked "upsampling" and "skip connection" each contain K convolutional layers. If an upsampling or skip connection is made Bayesian, then all convolutional layers therein are Bayesian.

3.2.2 Multi-model ensemble

The idea behind the multi-model ensemble (MME) is simple: train many point-prediction NNs, each with a different random seed, then ensemble the predictions. The random seed determines how the NN weights are initialized, and different initializations lead to different solutions, the "solution" being the final set of weights.

3.2.3 Bayesian neural networks

Any NN can be made Bayesian by replacing traditional (point-prediction) layers with Bayesian layers; thus, BNNs are a highly flexible approach to UQ. A point-prediction NN learns a single value for each weight, but a BNN learns a full *distribution* for some weights, determined by fitting the parameters of a user-chosen canonical distribution. In this work and in common practice, the normal distribution is chosen, so the BNN must learn two values for each Bayesian weight: the mean and variance. It is unnecessary to make all layers Bayesian. For example, a popular approach is to make only the last few layers Bayesian, which often achieves the same performance (*i.e.*, quality of mean predictions and uncertainty estimates) at a fraction of the computing cost (Jospin et al., 2022; Hertel et al., 2023).

While simple in theory, "making a layer Bayesian" is non-trivial in practice. To update a Bayesian weight w, one must compute the posterior distribution $p(w \mid \mathcal{D})$, where

$\mathcal{D}$ represents the training data. Solving the posterior exactly involves a computationally intractable integral, so in practice, variational inference is often used as an approximation (Hoffman et al., 2013; Ranganath et al., 2014; Rezende et al., 2014). Furthermore, weights in a NN are updated via gradient descent with backpropagation, which involves the gradient of the loss with respect to every weight, $\frac{\partial \mathcal{L}}{\partial w}$. However, a Bayesian weight is a random variable, and it is impossible to backpropagate the gradient through random variables. There are two popular solutions to this problem: the reparameterization trick (Kingma & Welling, 2013), which involves loss gradients with respect to only the *parameters* of the weight distribution (*e.g.*, the mean and variance of a normal distribution), and flipout (Wen et al., 2018), which involves sampling weight perturbations. The advantage of reparameterization is speed – per training epoch, it is faster than flipout – while the advantage of flipout is more accurate gradient estimates. This accuracy often translates to needing fewer training epochs, which can make flipout faster per network even though it is slower per epoch.

4 Experimental setup

We attempt five UQ methods with the U-net++ base model: CRPS-only, MME-only, MME/CRPS, BNN-only, and BNN/CRPS. Each of these methods generates an ensemble; for fair comparison across UQ methods, we set the ensemble size to 50. (Larger ensemble sizes lead to memory issues for training the BNN/CRPS models.) Specifically, we use the following techniques:

1. CRPS-only. Train a single U-net++ with the probabilistic loss function (Equation 6) and 50 output neurons per target variable ($N = 50$ in Figure 4).
2. MME-only. Train 50 U-net++ models, each with the deterministic loss function (Equation 4) and 1 output neuron per target variable.
3. MME/CRPS. Train 50 U-net++ models, each with the probabilistic loss function and 25 output neurons per target variable. Hence, the inner ensemble size is 25 and the outer ensemble size is 50 – leading to a total ensemble size of 1250, from which we randomly select 50 members.
4. BNN-only. Train a single Bayesian U-net++ with the deterministic loss function and 1 output neuron per target variable. At inference time, run the Bayesian U-net++ 50 times to get 50 predictions per target variable.
5. BNN/CRPS. Train a single Bayesian U-net++ with the probabilistic loss function and 50 output neurons per target variable. At inference time, run the Bayesian U-net++ 10 times, so that each probabilistic weight is sampled 10 times. Hence, the inner ensemble size is 50 and the outer ensemble size is 10 – leading to a total ensemble size of 500, from which we randomly select 50 members.

For methods involving a BNN, this leaves the question of which layers are Bayesian (probabilistic weights) and which are not (deterministic weights), as well as which method to use for training Bayesian layers (reparameterization or flipout). For both the BNN-only and BNN/CRPS methods, we optimize these hyperparameters with an experiment documented in Supplemental Section 2.

Models trained with a single UQ method can capture only one type of uncertainty (aleatory for CRPS-only, epistemic for MME-only and BNN-only), while those trained with a hybrid method can capture both types of uncertainty. Since uncertainty arising from out-of-sample data is partly aleatory and partly epistemic, we expect the hybrid UQ methods to perform best on the perturbed (validation and testing) data.

Experiment 1: Clean training data

In this experiment we train the models with clean (unperturbed) data, then task them with generalizing – both point predictions and uncertainty estimates – to perturbed validation and testing data. We expect all UQ methods to perform poorly on the perturbed data, because the perturbations are made along basis vectors with much less variability in the clean training data (Supplemental Figures S5-S9).

Experiment 2: Lightly perturbed training data

The confirmation of the above expectation (see Section 5) motivates another science question: what happens if the models are trained with *lightly perturbed*, instead of clean, data? Said differently, what happens if the models "see" a light version of the perturbations occurring in the validation and testing data? On the out-of-sample validation and testing data, we expect models trained with lightly perturbed data to perform better than models trained with clean data, but *how much* better is an open question.

Tools for evaluating UQ results

This section provides a light background on UQ-evaluation tools (both graphics and single-number metrics), which should be sufficient for readers to understand the ensuing results and discussion. See H23 for more details.

Figure 5 demonstrates our evaluation tools for two synthetic datasets. The first dataset (Figure 5a) represents a model with good mean predictions but too much spread (*i.e.*, ensemble ranges are wider than necessary); we call this Model A. The second dataset (Figure 5b) represents a model with poor mean predictions and too little spread (*i.e.*, the observation often falls completely outside the ensemble range); we call this Model B. The attributes diagram – which is a reliability curve with extra information (Hsu & Murphy, 1986) – is used to evaluate point predictions, showing the mean observation y_{true} as a function of the ensemble-mean prediction $\overline{y_{\text{pred}}}$. This graphic is used to assess conditional bias, *i.e.*, bias as a function of $\overline{y_{\text{pred}}}$. Model A has no conditional biases (Figure 5c), leading to a reliability curve that follows the 1:1 line and a reliability (REL, the mean squared distance between the curve and 1:1 line) of 0.00 K^2 day^{-2}. Meanwhile, Model B completely misses the extremes, *i.e.*, the lowest (highest) predictions are far too high (low). This leads to the classic sigmoid-shaped reliability curve and a large REL (Figure 5d).

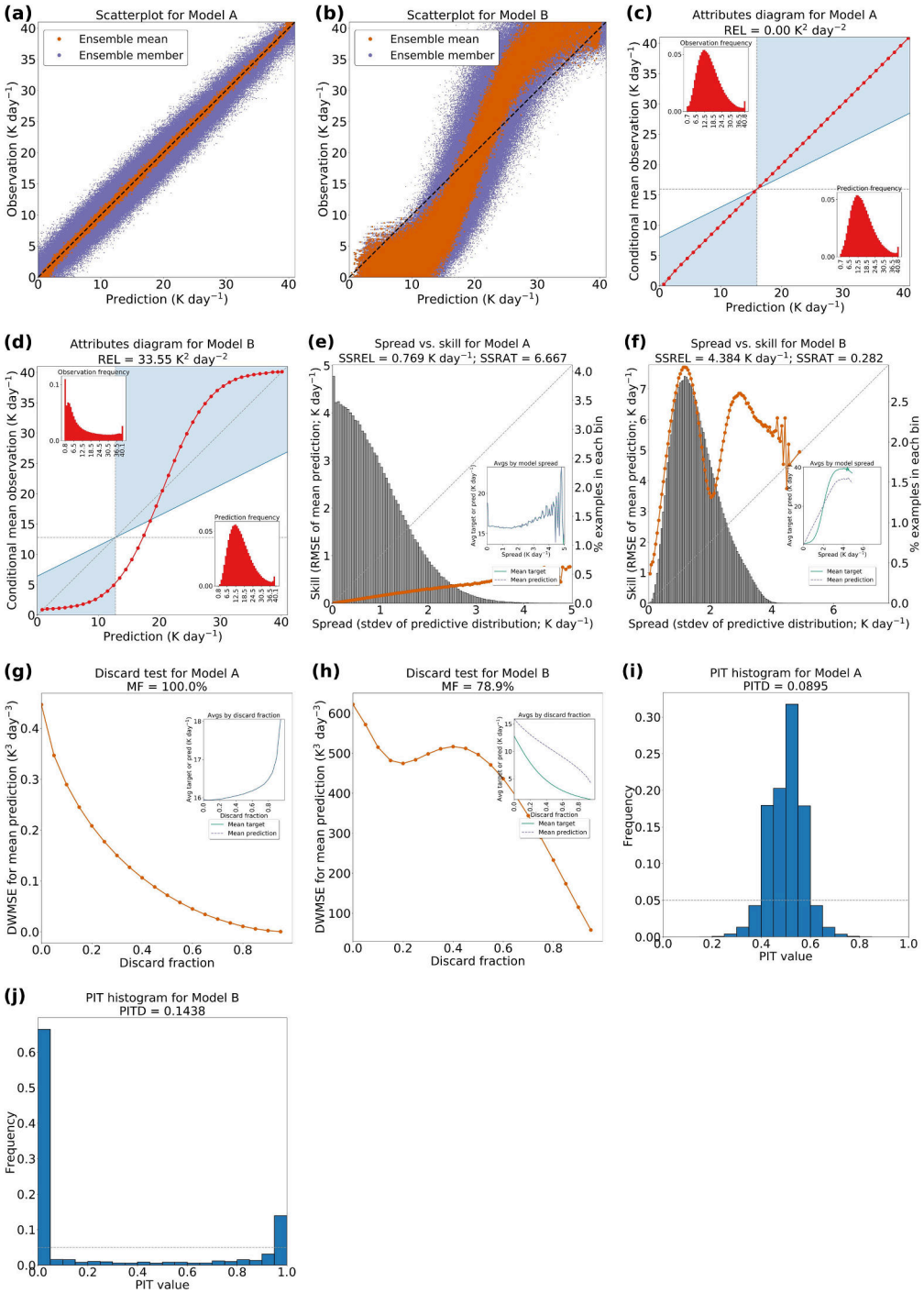


Figure 5: Demonstration of evaluation tools on two synthetic datasets. [a-b] The two synthetic datasets, representing "Model A" and "Model B". [c] Attributes diagram for Model A. Of the dashed grey lines: the diagonal (1:1) line represents the perfect reliability curve; the vertical line is the climatology line, representing the mean observation in the dataset (16 K day^{-1}); and the horizontal line is the no-resolution line, representing the reliability curve for a completely uninformative model. The blue shading is the positive-skill area, where the model's MSE is better than that yielded by always predicting climatology (here, 16 K day^{-1}). [d] Same but for Model B. [e] Spread-skill plot for Model A. The diagonal (1:1) line represents a perfect spread-skill curve; the grey histogram shows how often each spread value occurs; and the inset plot shows any biases as a function of model spread. [f] Same but for Model B. [g] Discard test for Model A. The inset plot shows any biases as a function of discard fraction. [h] Same but for Model B. [i] PIT histogram for Model A. The dashed line represents a perfect (uniform) PIT histogram. [j] Same but for Model B.

Remaining tools shown in Figure 5 are for uncertainty estimates rather than point predictions. The spread-skill plot (Delle Monache et al., 2013) shows the root mean square error (RMSE) achieved by $\overline{y_{\text{pred}}}$, or "skill," as a function of the ensemble standard deviation, or "spread". For a perfect model, the spread-skill ratio is 1.0 across all spread values, so the curve follows the 1:1 line. Model A (Figure 5e) is very overspread or "underconfident," leading to a curve well below the 1:1 line and a large overall spread-skill ratio (SSRAT). Model B (Figure 5f) has the opposite problem. Spread-skill reliability (SSREL), the mean distance between the curve and 1:1 line, is substantially lower (better) for Model A.

In the discard test, data samples are thrown out in descending order of model uncertainty (*i.e.*, the highest-uncertainty samples are thrown out first) and the effect on model error is observed. The error should decrease monotonically, *i.e.*, whenever the discard fraction increases. For all discard tests in this paper, model error is based on the ensemble mean $\overline{y_{\text{pred}}}$ and model uncertainty is the height-averaged variance of HR predictions. (Mathematically, this is $\frac{1}{H}\sum_{h=1}^{H}\left[\frac{1}{N-1}\sum_{i=1}^{N}(\hat{r}_{hi}-\overline{\hat{r}_h})^2\right]$, where $\overline{\hat{r}_h}$ is the ensemble mean at the h^{th} height; all other variables are defined in Equation 5. There are two reasons that we use only HR, and not flux, to define overall uncertainty. First, most of the model's outputs [127 of every 130] are HRs; second, combining HR and flux uncertainties into an overall uncertainty is non-trivial, as they have different units.) Model A (Figure 5g) has a perfect discard test, leading to a monotonicity fraction (MF) of 100%. Model B (Figure 5h) has an imperfect discard test; error increases as the discard fraction increases from 20-25%, from 25-30%, from 30-35%, and from 35-40%. Thus, model error decreases only 15 of 19 times that the discard fraction increases, leading to an MF of $\frac{15}{19} = 78.9\%$.

The probability integral transform (PIT), defined for each data sample, is the ranking of y_{true} in the distribution $\vec{y}_{\text{pred}}$. For example, if y_{true} is less than all $\vec{y}_{\text{pred}}$, its PIT is 0.0; if y_{true} is greater than all $\vec{y}_{\text{pred}}$, its PIT is 1.0; if y_{true} is the median of all $\vec{y}_{\text{pred}}$, its PIT is 0.5; etc. The PIT *histogram* – which is similar to the rank histogram, or "Talagrand diagram" (Hamill, 2001), and can be interpreted similarly – then shows the distribution of PIT values. For a perfectly calibrated model – which is neither overconfident nor underconfident – all PIT values occur equally often, leading to a uniform histogram. For Model A (Figure 5i), nearly all PIT values are between 0.3 and 0.7, meaning that y_{true} is usually in the middle 40% of the $\vec{y}_{\text{pred}}$ distribution and rarely at the extremes. In other words, the $\vec{y}_{\text{pred}}$ distribution is usually too wide; the model is underconfident. For Model B (Figure 5j), nearly all PIT values are below 0.05 or above 0.95, meaning that y_{true} is usually in the bottom or top 5% of the distribution. In other words, the $\vec{y}_{\text{pred}}$ distribution is usually too narrow; the model is overconfident. The PIT deviation (PITD), the mean absolute difference between bar height and the horizontal line, is substantially better (lower) for Model A. (The horizontal line line denotes the bar height for the ideal [uniform] PIT histogram, which is $\frac{1}{\text{number of bins}}$.)

REL, SSREL, and PITD are negatively oriented with a perfect value of 0.0; MF is positively oriented with a perfect value of 1.0; and the perfect SSRAT is 1.0, with higher (lower) values indicating underconfidence (overconfidence).

Lastly, we define "large point error" as a data sample where $\overline{y_{\text{pred}}}$ has absolute error ≥ 1 K day^{-1}; we define "catastrophic error" as a large point error where the observation also falls outside the 95% confidence interval (in other words, PIT is either < 0.025 or > 0.975).

How to read the results

Sections 5 and 6, which discuss the results of the two experiments, contain many specific terms from the RT application and the field of UQ. All these terms have been

explained briefly heretofore, with longer explanations found in L23 (for RT) and H23 (for UQ). However, we do not expect readers to be fluent in these terms, so we highlight "key points" throughout Sections 5 and 6. Readers with less interest in the details can jump directly to the key points.

5 Results for Experiment 1: Clean training data

We start with overall diagnostics (metrics computed from the entire validation or testing set), which allow us to understand the UQ methods' performance and choose the best method. Then we present a small number of case studies, which allow us to understand the UQ methods' performance in a way that overall diagnostics cannot.

5.1 Overall diagnostics

Figure 6 compares all five UQ methods on the *moderately perturbed* validation data. Error metrics pertaining to heating rates (HR) are averaged over the 127 heights; those pertaining to fluxes are averaged over the three variables ($F_{\text{down}}^{\text{sfc}}$, $F_{\text{up}}^{\text{TOA}}$, and F_{net}).

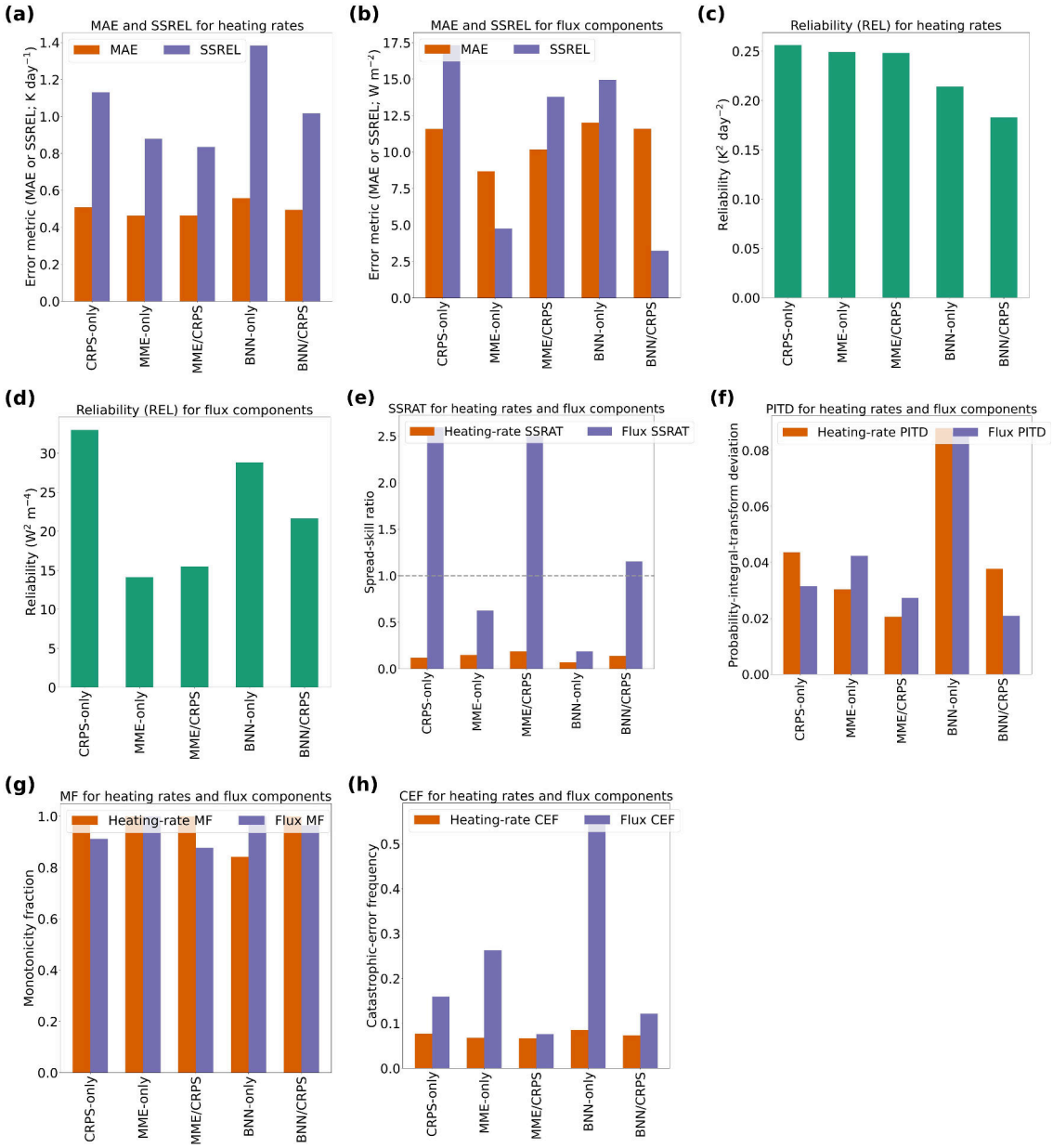


Figure 6: Comparison of UQ methods on validation data, for models trained with clean data. In panel g, higher is better; in panel e, closer to 1.0 is better; in all other panels, lower is better. "CEF" in panel h is catastrophic-error frequency.

We highlight several observations from Figure 6:

1. UQ methods that can capture only epistemic uncertainty – *i.e.*, MME-only and BNN-only – produce too little spread for both HR and fluxes (panel e). This suggests that much of the uncertainty in the validation data is aleatory.
2. Methods that can capture aleatory uncertainty – *i.e.*, those involving the CRPS – produce too much spread for fluxes (panel e). However, out of these three methods, BNN/CRPS is the least overspread.

3. All UQ methods produce far too little spread for HR (panel e); this is true at all 127 heights (not shown).
4. All UQ methods produce catastrophic errors at least ∼10% of the time (panel h); the non-hybrid methods (CRPS-only, MME-only, and BNN-only) produce catastrophic errors substantially more often.
5. The BNN/CRPS method performs best on 4 of the 10 uncertainty-based metrics (flux SSREL, flux SSRAT, flux PITD, and HR MF). The MME/CRPS method performs best on 6 of the 10 uncertainty-based metrics (HR SSREL, HR SSRAT, HR PITD, HR MF, HR CEF, and flux CEF), where CEF is catastrophic-error frequency. However, MME/CRPS performs worst on flux MF (panel g) and second-worst on flux SSRAT (panel e), while BNN/CRPS does not perform this badly for any uncertainty-based metric.
6. MME/CRPS outperforms BNN/CRPS on 3 of the 4 point-prediction-based metrics (HR MAE, flux MAE, and flux REL; not HR REL). Thus, MME/CRPS produces better point predictions than BNN/CRPS; however, the differences here are small.

Key points: Points 3 and 4 exemplify that, when trained with clean data, all UQ methods fail dramatically. Based on points 5 and 6, we judge that the best UQ method is BNN/CRPS, followed by MME/CRPS. Both are hybrid methods, which can capture both aleatory and epistemic uncertainty. The remaining analysis will focus largely on BNN/CRPS.

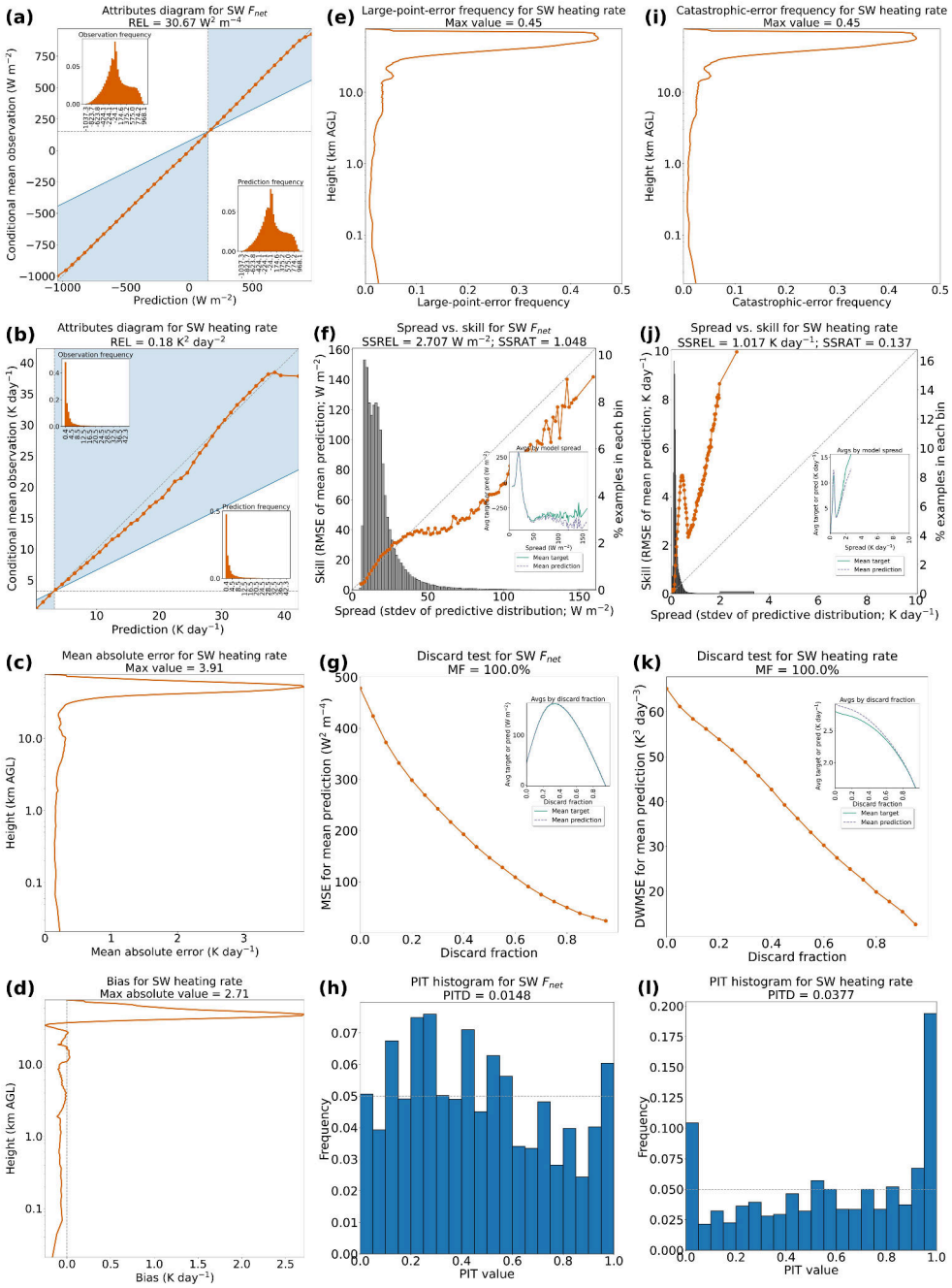


Figure 7: Detailed results of the BNN/CRPS method on validation data, for a model trained with clean data. [a-e] Evaluation of point predictions (ensemble means). [a] Attributes diagram for F_{net}; [b] attributes diagram for HR, aggregated over all heights; [c] profile of mean absolute errors for HR; [d] profile of mean signed errors (biases) for HR; [e] profile of large-point-error frequencies for HR. [f-h] Evaluation of uncertainty estimates for F_{net}. [f] Spread-skill plot; [g] discard test; [h] PIT histogram. [i-l] Evaluation of uncertainty estimates for HR. [i] Profile of catastrophic-error frequencies for HR; [j-l] as in panels f-h but for HR.

Figure 7 shows detailed results for the BNN/CRPS model, based on the *moderately perturbed* validation data. For both F_{net} (panel a) and HR (panel b), the attributes diagram is nearly perfect, except a large positive bias ($\sim$5 K day^{-1}) when $\overline{y_{\text{pred}}} \gtrsim 38$ K day^{-1}. In other words, the highest HR predictions are too high. Panels c-d show MAE and bias at each height for the ensemble-mean HR prediction. For shortwave RT, errors on the order of 0.1 K day^{-1} are generally considered acceptable – *e.g.*, Table 2 of Krasnopolsky et al. (2012), page 7 of Song and Roh (2021), Figure 1 of Kim and Song (2022). At most heights the errors are on this order, except in the upper stratosphere, where MAE jumps to 3.91 K day^{-1} and bias jumps to 2.71 K day^{-1}. Panel e shows the frequency of large point errors for HR, which is below 5% throughout the troposphere but jumps to 45% in the upper stratosphere. Error maxima in the upper stratosphere are associated with perturbed ozone layers; some examples will be shown in case studies.

Figures 7f-h show the quality of uncertainty estimates for F_{net}. The spread-skill plot (panel f) shows that F_{net} predictions are almost perfectly calibrated when spread $\lesssim 40$ W m^{-2}; for higher spread values, the model is slightly underconfident. The discard test (panel g) shows that, despite the underconfidence at higher spread values, the model's overall uncertainty is strongly correlated with its error for F_{net}. In other words, one can trust that lower uncertainty means lower expected F_{net} error. The PIT histogram (panel h) shows that the model's F_{net} predictions are quite well calibrated, except slightly too many PIT values below 0.5. In other words, y_{true} falls in the bottom half of the $\vec{y}_{\text{pred}}$ distribution more often than it should. Meanwhile, Figures 7i-l show the quality of uncertainty estimates for HR. Panel i shows the profile of CEFs, which are similar to large-error frequencies (panel e). In other words, most large errors are also catastrophic errors, because the confidence interval (CI) cannot account for errors > 1 K day^{-1}. The spread-skill plot (panel j) shows that the model is extremely overconfident, producing only 14% as much spread as it should. The discard test (panel k) shows that, despite this overconfidence, the model's overall uncertainty is strongly correlated with its HR error. Finally, the PIT histogram (panel l) shows that the model's HR predictions are poorly calibrated, with extreme PIT values (the first and last bars) occurring for 30% of data samples. In other words, y_{true} falls near the bottom or top of the $\vec{y}_{\text{pred}}$ distribution 30% of the time, three times as often as it should.

Having used the *moderately perturbed* validation data to select the best UQ method (BNN/CRPS), we now evaluate BNN/CRPS on the *heavily perturbed* testing data. By comparison of Figure 6 and Supplemental Figure S23, the overall ranking of UQ methods is similar between the validation and testing data. Most importantly, BNN/CRPS appears to be the best method for both datasets.

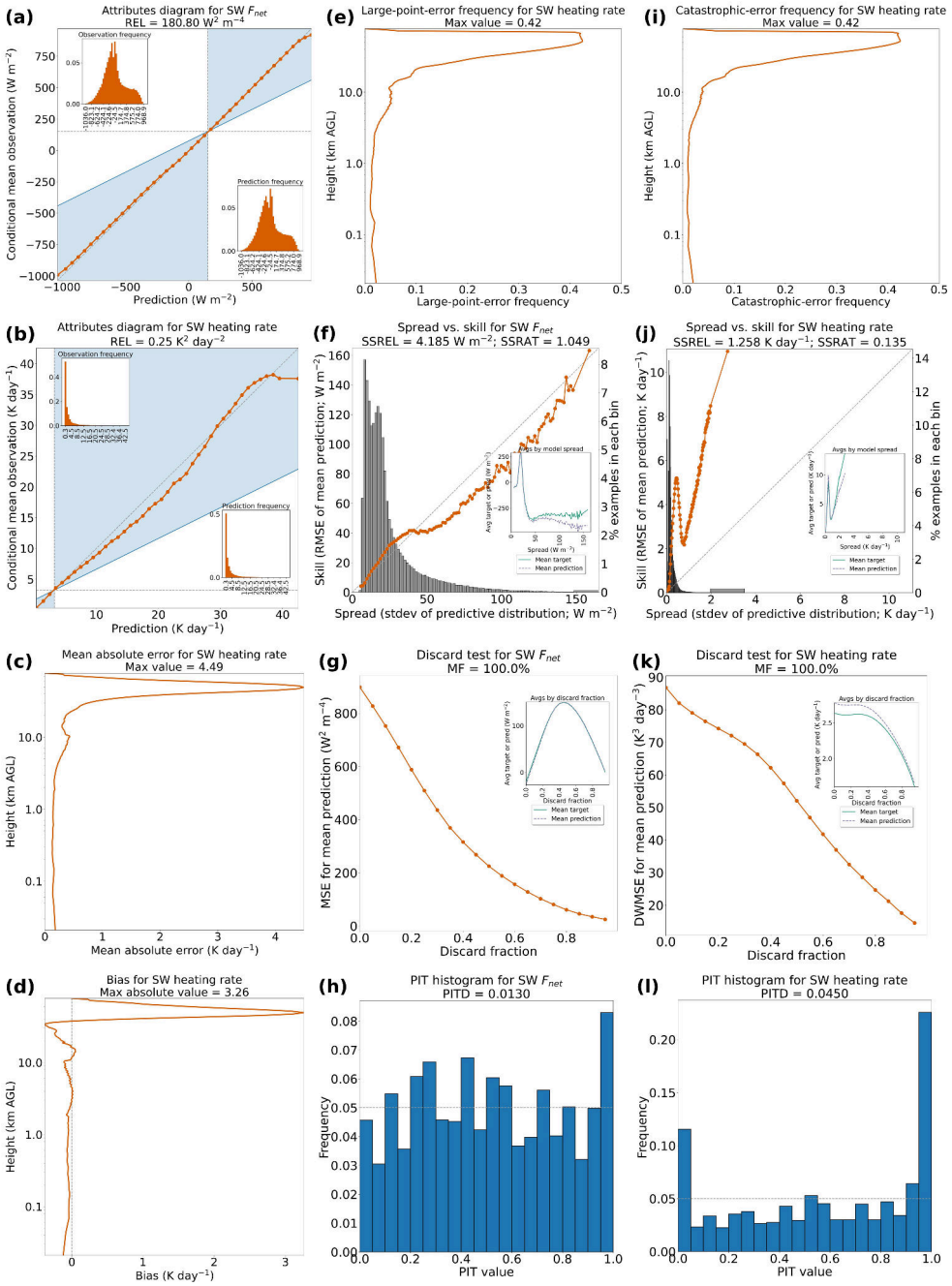


Figure 8: Detailed results of the BNN/CRPS method on testing data, for a model trained with clean data. Formatting is explained in the caption of Figure 7.

Figure 8 shows detailed results for the BNN/CRPS model on the testing data. Here we highlight differences from the validation results (Figure 7). The attributes diagrams (panels a-b) show that point predictions of F_{net} and HR are worse on the testing data (note the higher REL values), but these plots still indicate good skill except for the highest HR predictions. The MAE and bias profiles (panels c-d) show that point predictions of HR are still mostly acceptable in the troposphere, but problems in the stratosphere are worse in the testing data. Large-error frequencies for HR (panel e) are similar to the

validation data but with a slightly better maximum (7% decrease) in the upper stratosphere. Results for F_{net} uncertainty (panels f-h) are similar to the validation data, except with a worse SSREL (55% increase) and slightly better PITD (12% decrease). Results for HR uncertainty (panels i-l) are also similar to the validation data – showing very poor skill – except with a slightly better maximum for CEF (7% decrease), worse SSREL (24% increase), and worse PITD (19% increase).

Key points: For models trained with clean data, even the best model produces unacceptable errors, as expected. The most notable errors are poor HR predictions in the stratosphere, with CEF $> 40\%$; poor HR predictions at the highest values, with a bias of $\gg 1$ K day^{-1}; and poor HR uncertainty estimates throughout the atmosphere, with SSRAT $< 14\%$. Results for the testing data are worse than for the validation data, as expected.

5.2 Case studies

Case study 1: Validation data. Figure 9 shows a case with the following perturbations: a two-layer ice cloud (panel a), a multi-layer liquid cloud with large/noisy LWC values (panel a), and an ozone layer with noisy mixing ratios (panel b). For the HR spike due to ice cloud (around 10 km), all point predictions are too low and most CIs (for all models except BNN/CRPS; panel f) miss the observation. For the HR spikes due to liquid cloud (from 1-3 km), point predictions have a large error (> 1 K day^{-1}) but observations generally fall within the CI, especially for the non-BNN models (panels c, d, g). For the HR spike due to ozone (around 45 km), all point predictions and CIs are far too low – *i.e.*, all models produce a catastrophic error. The models also fail to capture other aspects of ozone-related heating (from 15-60 km), including the HR minimum around 60 km.

Key points: This case study exemplifies that while perturbed cloud sometimes causes larger point errors than perturbed ozone, perturbed ozone causes catastrophic errors more often. This conclusion is supported more rigorously by comparing panel i across Supplemental Figures S26-S28. The reason is that ozone varies much less in the training data than LWC/IWC – *e.g.*, all training samples have exactly one ozone layer with a maximum mixing ratio between 5.5 and 18.5 mg kg^{-1}, while different training samples have very different LWC/IWC profiles. Thus, perturbed ozone layers are more alien to the training data than perturbed cloud layers.

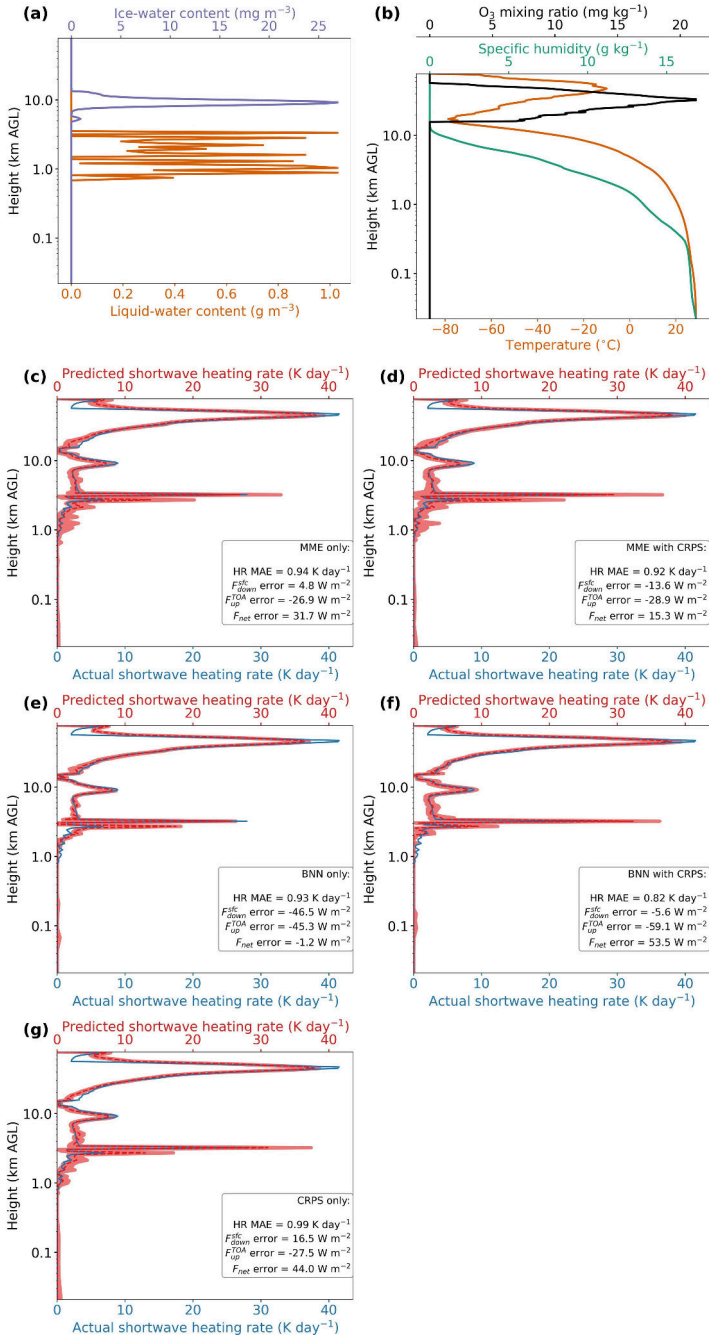


Figure 9: Case study for models trained with clean data and applied to a validation sample: 0000 UTC 9 Dec 2020, 1.58 °S, 94.80 °E. [a-b] Key predictor variables, *i.e.*, those subject to perturbation. [c] Actual HR profile (blue), along with ensemble-mean prediction (dashed red line) and 95% confidence interval (shaded red envelope), from the MME-only model. [d] Same but for MME/CRPS model. [e] Same but for BNN-only model. [f] Same but for BNN/CRPS model. [g] Same but for CRPS-only model. In the legends, "HR MAE" is the MAE of ensemble-mean HR predictions over the 127 heights; "F_{net} error" is the ensemble-mean F_{net} prediction minus actual; and "$F^{\text{sfc}}_{\text{down}}$ error" and "$F^{\text{TOA}}_{\text{up}}$ error" are defined analogously.

Case study 2: Cloudy testing data. Figure 10 shows a case with two following perturbations: a shallow ozone layer and near-surface moist layer (panel b). All models struggle with ozone-related heating (from 15-30 km), as in the validation case but worse. The near-surface moist layer (bottom 0.6 km) causes an HR maximum of $\sim$10 K day^{-1}, for which all UQ methods fail completely. The best models in this region are MME-only (panel c) and MME/CRPS (panel d), but the HR maximum is still $\sim$3 K day^{-1} above both ensemble means and $\sim$1 K day^{-1} above both CIs, so these errors are considered catastrophic.

Key points: This case study exemplifies two conclusions from the broader dataset. First, although perturbed near-surface moisture generally causes smaller errors than perturbed ozone and cloud layers, near-surface moisture can still cause catastrophic errors. These are most common in profiles with little to no cloud, leaving ample solar radiation to reach the near-surface and interact with water vapour there. Second, perturbations in the testing data cause worse errors than perturbations in the validation data (*cf.* Figures 7 and 8), as expected.

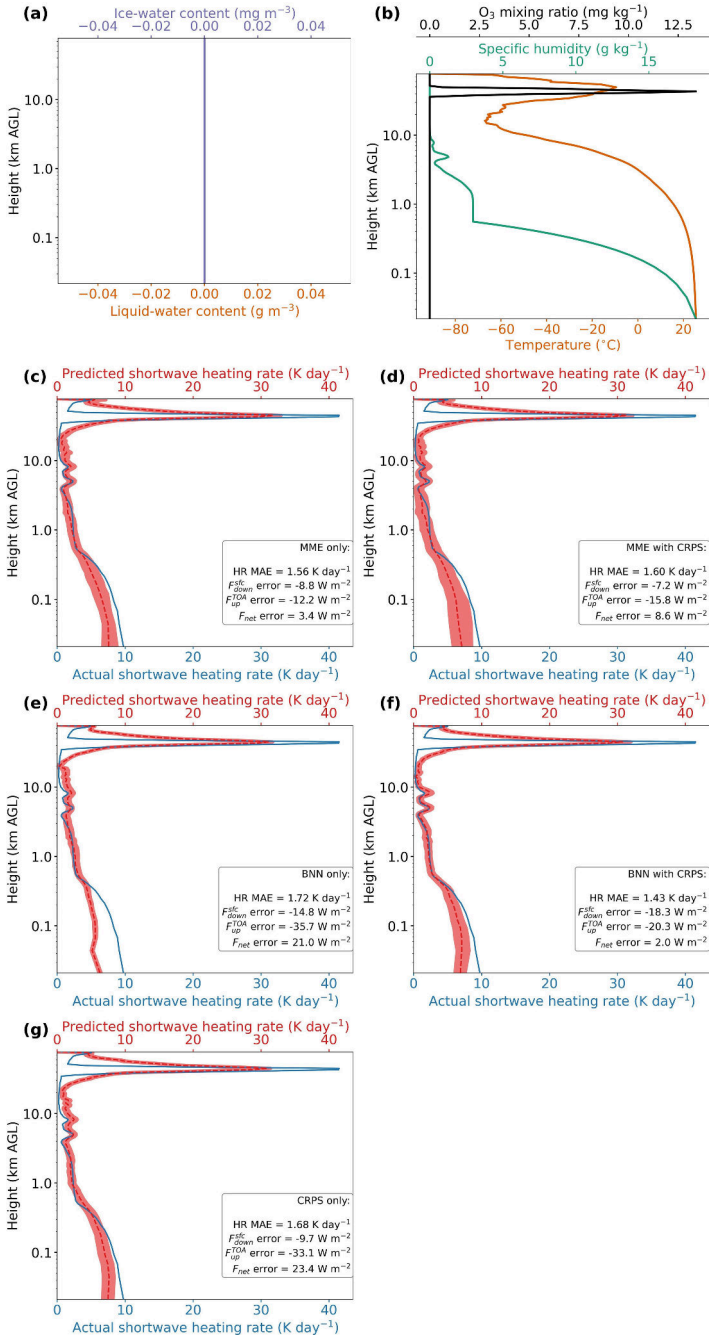


Figure 10: Case study for models trained with clean data and applied to a testing sample: 1200 UTC 18 Dec 2020, 27.47 °N, 6.91 °E. All formatting is explained in the caption of Figure 9.

Case study 3: Cloud-free testing data. Figure 11 shows a case with the following perturbations: a shallow ozone layer with large mixing ratios (panel b), three cloud layers with large/noisy LWC values (panel a), a near-surface moist layer with specific humidity reaching ∼30 g kg^{-1} (panel b), and a near-surface warm layer with temperature reaching ∼35 °C (panel b). There is virtually no heating near the surface, because all solar radiation is attenuated by the clouds above. The models capture this lack of heat-

ing quite well, except for MME/CRPS (panel d) and BNN/CRPS (panel f), which produce an erroneous HR maximum in the bottom 0.2 km. For the HR spike due to liquid cloud (around 10 km), all models produce a catastrophic error. This error is worse than cloud-related errors in the validation case (Figure 9), consistent with the more extreme LWC values in this, a testing case. For ozone-related heating, all models produce catastrophic errors throughout the stratosphere.

Key points: The models were trained with clean data and simply have not *seen* liquid cloud or ozone layers like the one here (*cf.* Figure 11b and Supplemental Figure S9a). Thus, when presented with *heavily perturbed* testing data, which are far out of sample compared to the training data, the models (including their uncertainty estimates) completely fail. This confirms our expectation from Section 4 and motivates Experiment 2 – with lightly perturbed, instead of clean, training data.

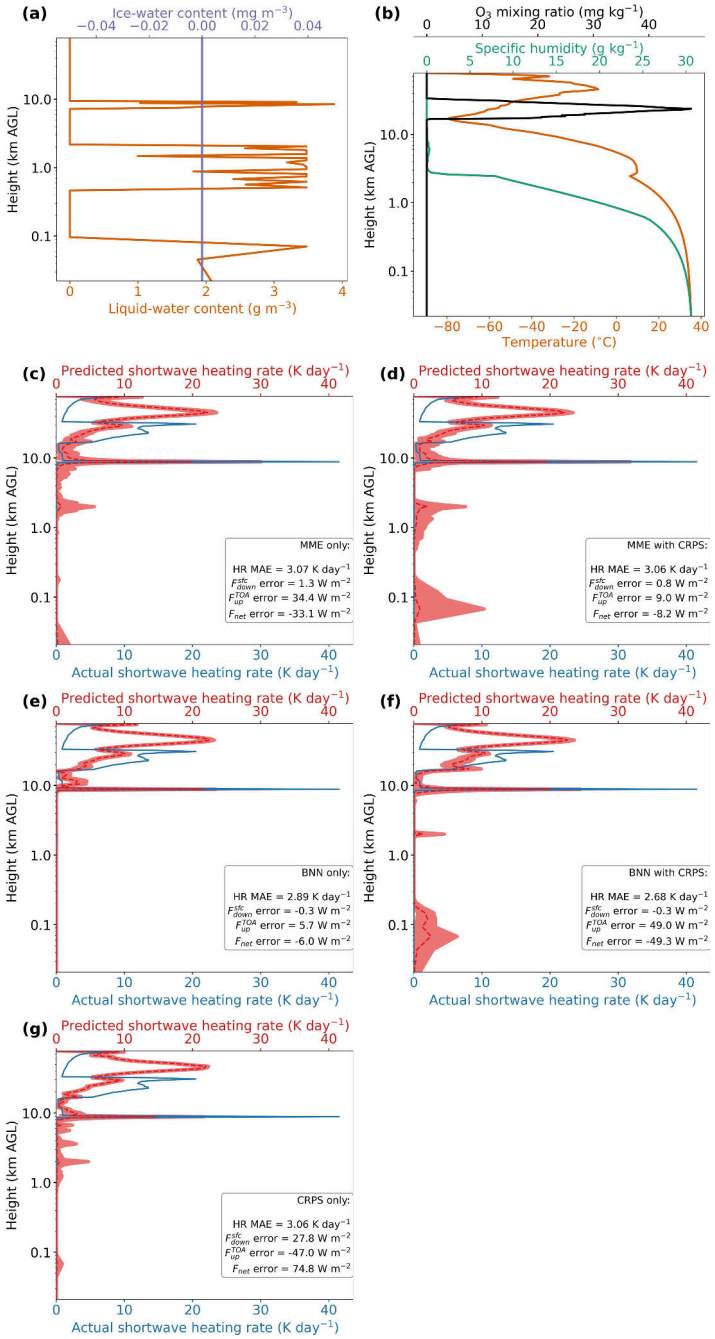


Figure 11: Case study for models trained with clean data and applied to a testing sample: 1200 UTC 12 Aug 2020, 16.69 °S, 37.14 °E. All formatting is explained in the caption of Figure 9.

6 Results for Experiment 2: Lightly perturbed training data

As in the discussion for Experiment 1, we start with overall diagnostics, then dig deeper with case studies.

6.1 Overall diagnostics

Figure 12 compares all five UQ methods on the validation data. This figure, which shows models trained with lightly perturbed data (henceforth LP-trained models), is analogous to Figure 6, which shows clean-trained models. We highlight several observations from Figure 12. Unless otherwise stated, these observations are true for the clean-trained models as well.

1. MME-only and BNN-only produce too little spread for all variables (panel e).
2. Methods involving the CRPS produce too much spread for fluxes, but BNN/CRPS is the least overspread among these methods (panel e).
3. The clean-trained models produce *far* too little spread for HR (no SSRAT > 0.19; Figure 6e). However, among the LP-trained models, all except BNN-only produce an HR SSRAT > 0.74 (Figure 12e).
4. The LP-trained models produce far fewer catastrophic errors than the clean-trained models, especially for HR (panel h). For example, the LP-trained model with the MME/CRPS method produces a CEF of 0.26% and 4.2% for HR and flux, respectively; analogous values for the clean-trained model are 6.7% and 7.6%.
5. As for the clean-trained models, BNN/CRPS produces the best uncertainty estimates overall (performing best on 8 of 10 uncertainty-based metrics). Unlike for the clean-trained models, there is no clear second-best method for uncertainty estimates.
6. BNN/CRPS also produces competitive point predictions (4^{th}-best HR MAE, 2^{nd}-best flux MAE, best HR REL, best flux REL).

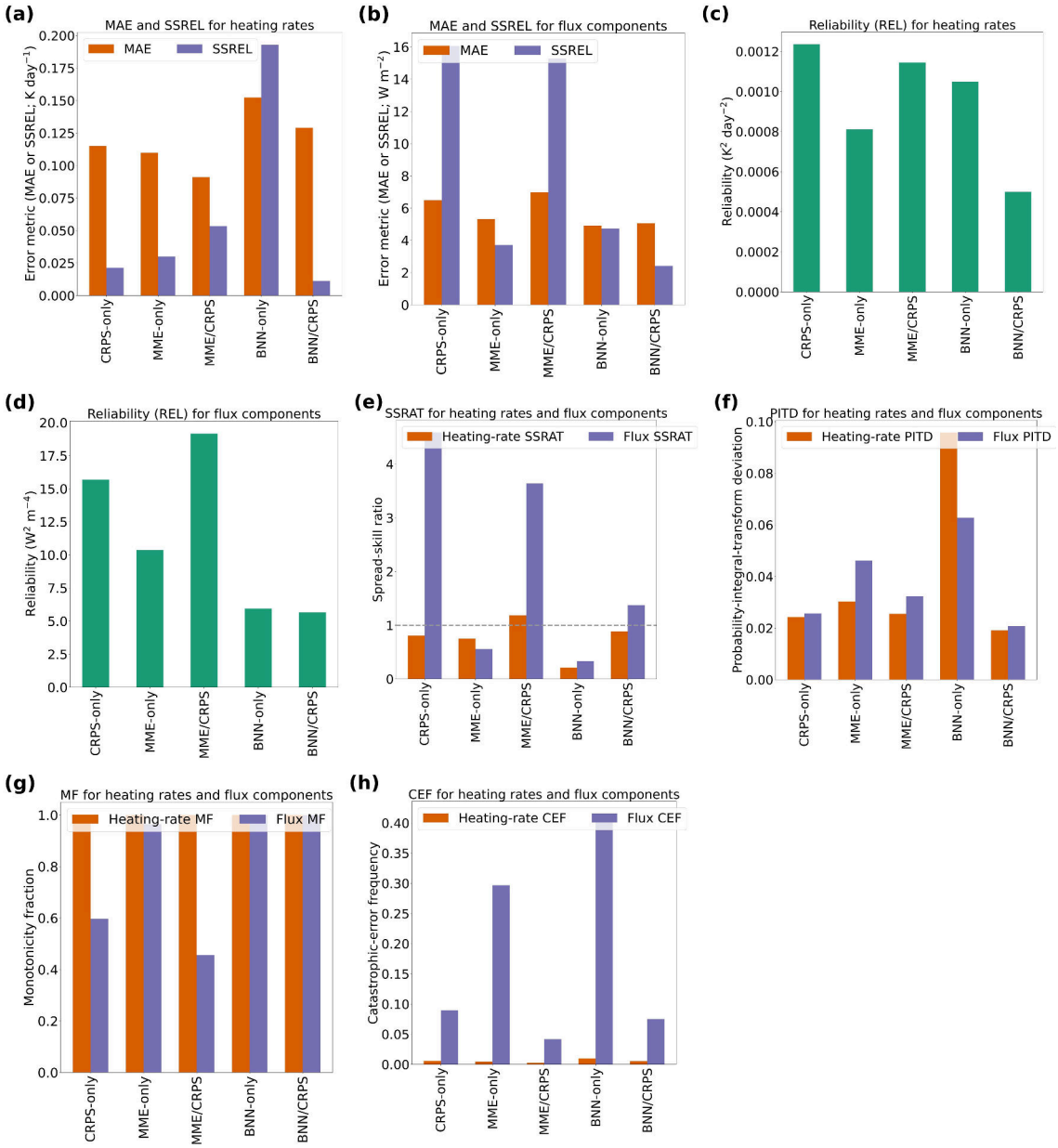


Figure 12: Comparison of UQ methods on validation data, for models trained with lightly perturbed data. In panels a-d and f, lower is better; in panel e, closer to 1.0 is better; and in panel g, higher is better.

As for the clean-trained models, we judge that BNN/CRPS is the best UQ method overall. Supplemental Figure S29 shows that this conclusion also holds on the testing data. The remainder of this section focuses on the BNN/CRPS method and, for brevity, focuses on the testing data rather than the validation data. We have already seen for clean-trained models that performance deteriorates from the validation to the testing data, and this is true for LP-trained models as well.

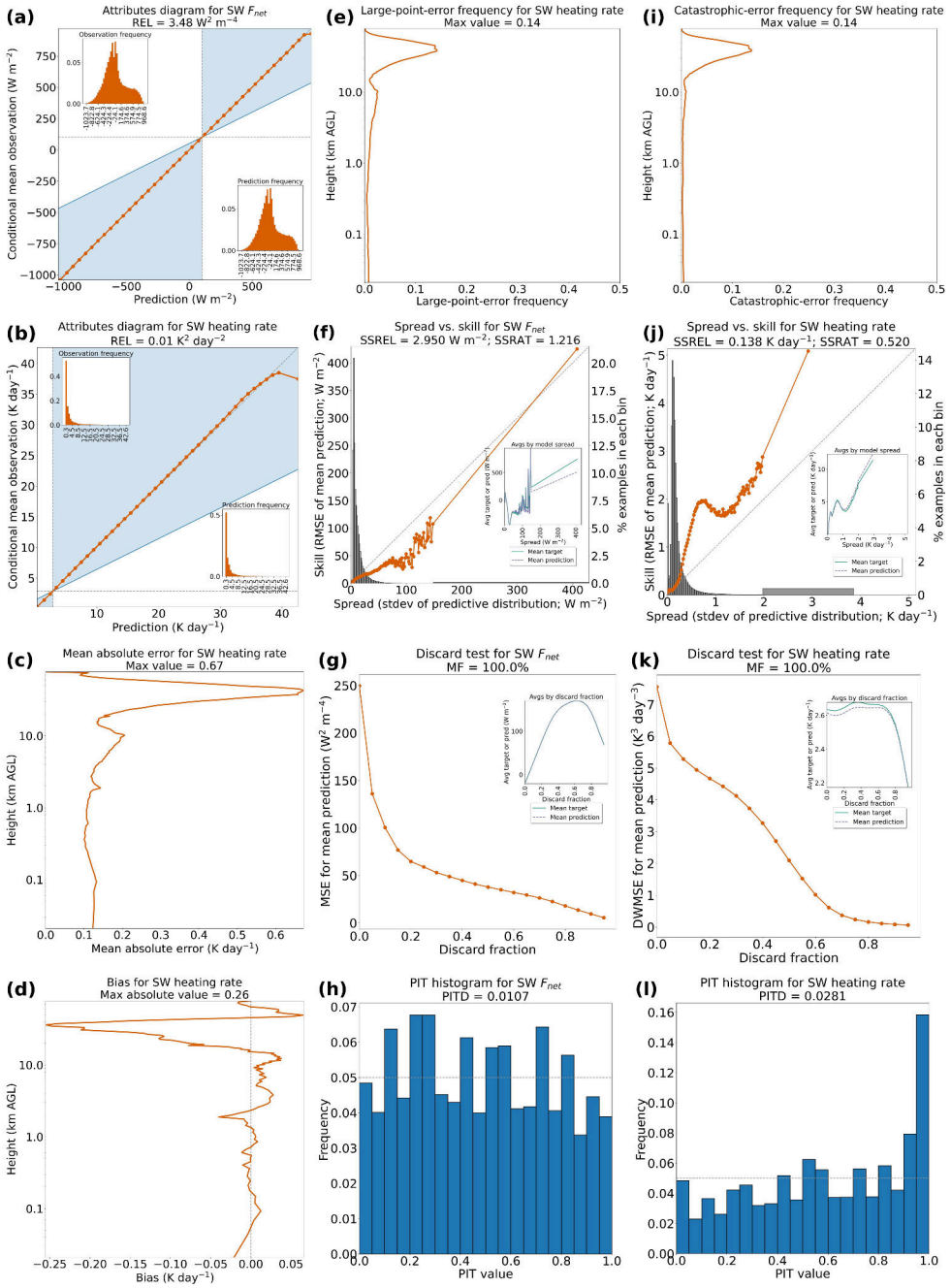


Figure 13: Detailed results of the BNN/CRPS method on testing data, for a model trained with lightly perturbed data. Formatting is explained in the caption of Figure 7.

Figure 13 shows detailed testing results for the LP-trained BNN/CRPS model. We compare these results to the clean-trained BNN/CRPS model (Figure 8). The attributes diagrams (panels a-b) show that point predictions of both F_{net} and HR are extremely well calibrated, except for the highest HR predictions. The attributes diagrams are better for the LP-trained model. The MAE and bias profiles (panels c-d) show that point HR predictions are better for the LP-trained model, *much* better in the stratosphere. Specifically, the maximum MAE is 85% lower, and the maximum absolute bias is 92%

lower. The frequency of large HR errors (panel e) is also better for the LP-trained model, with a 69% decrease in the upper-stratosphere maximum. Results for F_{net} uncertainty (panels f-h) show that the LP-trained model is well calibrated, although slightly underconfident in general (panel f) and producing slightly too many PIT values below 0.5 (panel h). The SSREL and PITD are better than for the clean-trained model, but the SSRAT is worse. Results for HR uncertainty (panels i-l) show that the LP-trained model is poorly calibrated, with overconfidence at nearly all spread values (panel j) and too many PIT values above 0.5 (panel l). However, these results are *much* better than for the clean-trained model, with a 69% decrease in maximum CEF, 89% decrease in SSREL, 285% increase in SSRAT, and 38% decrease in PITD. Overall, the comparison of Figures 8 and 13 shows that the LP-trained model is better than the clean-trained model, but three results of the LP-trained model are still concerning: large positive bias for HR point predictions $\gtrsim 38$ K day^{-1}, a 14% frequency of catastrophic HR errors in the upper stratosphere, and large overconfidence for HR in general. Supplemental Figure S30 – analogous to Figure 13 but for the validation data – shows that similar concerns exist in the validation data but are much less severe.

Key points: Experiment 1 showed that training with clean data, which barely sample important basis vectors in the validation/testing data, leads to catastrophic errors. Experiment 2 shows that perturbing the training data *just a little* along these basis vectors leads to much better performance on the validation/testing data, even if the latter are still out of sample. However, catastrophic errors still occur, showing that ML-UQ is not magic.

6.2 Case studies

Both case studies in this section are from the testing data.

Case study 1: Extreme liquid cloud and humidity. Figure 14 shows a case with the following perturbations: a very dense liquid cloud (panel a), an ozone layer with very large/noisy mixing ratios (panel b), and a near-surface moist layer with very large humidity (panel b). Also, the maximum ozone content occurs at a lower height than usual, around 20 km (in the lower stratosphere). For ozone-related heating around this level, all models produce a catastrophic error. However, for ozone-related heating above this level, most models (all except BNN-only; panel e) perform quite well. This result is in stark contrast to case studies for the clean-trained models, which struggle with perturbed ozone everywhere in the stratosphere. For the heating due to liquid cloud and the moist layer (from 0-0.3 km), only the MME-only and MME/CRPS models (panels c-d) perform well. The BNN-only and BNN/CRPS models (panels e-f) produce catastrophic errors, and the CRPS-only model (panel g) produces a large point error (~ 10 K day^{-1}) for the HR maximum.

Key points: Although the LP-trained models are much better than the clean-trained models, every LP-trained model produces a catastrophic error somewhere in the profile.

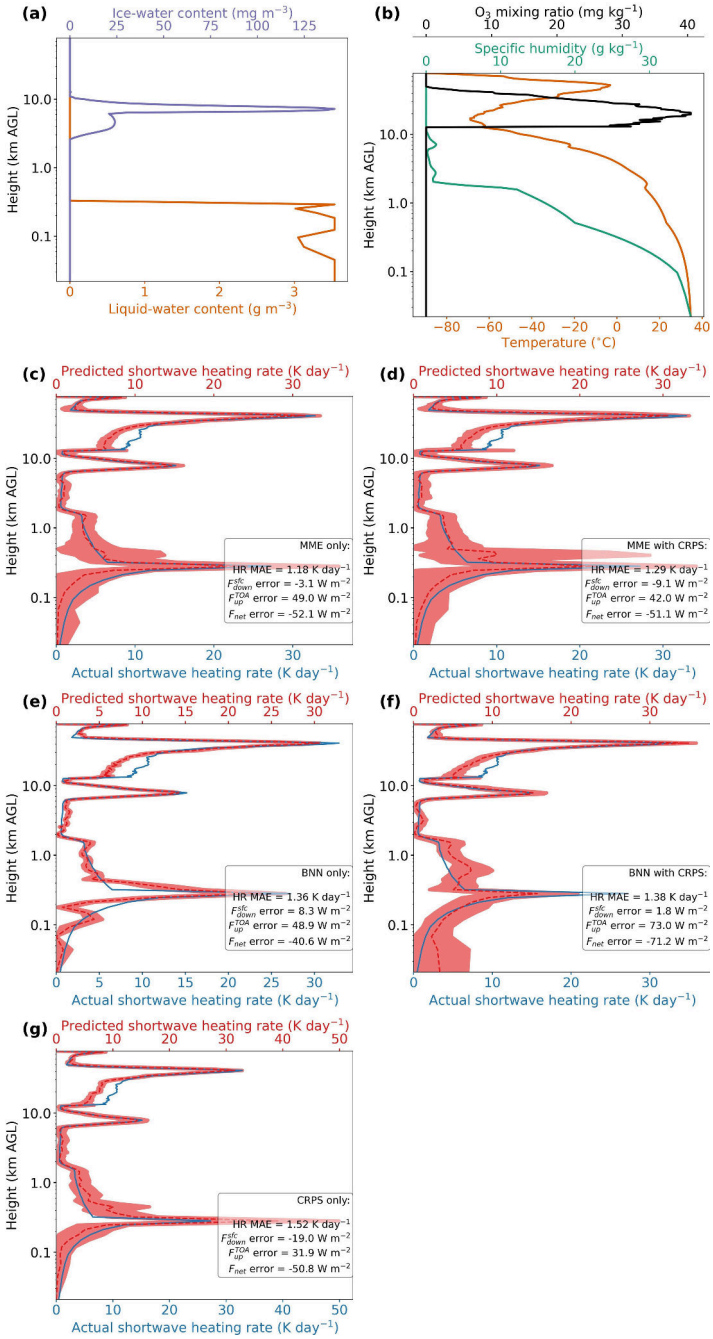


Figure 14: Case study for LP-trained models applied to a testing sample: 1800 UTC 22 Apr 2020, 30.99 °N, 99.26 °W. All formatting is explained in the caption of Figure 9.

Case study 2: Extreme ice cloud and ozone. Figure 15 shows a case with the following perturbations: a very dense two-layer ice cloud (panel a) and an ozone layer with small/noisy mixing ratios (panel b). There is virtually no heating below 7 km, because all solar radiation is attenuated by the ice cloud above; all models capture this lack of heating well. For the ice-cloud-related heating (around 8 km), all point predictions are 1-3 K day^{-1} too low. However, most CIs (for all models except BNN-only and BNN/CRPS; panels e-f) capture the observed HR. For the ozone-related heating (above 10 km), the

MME/CRPS model (panel d) performs best. Specifically, at every height except the HR maximum around 29 km, the point error is < 1 K day^{-1} and the CI captures the observation. The other models perform nearly as well, except the BNN-only model, which produces catastrophic errors from 30-45 km.

Key points: Again, the LP-trained models are much better than the clean-trained models. The MME/CRPS model even manages to produce no catastrophic errors for this case study, which is far out of sample.

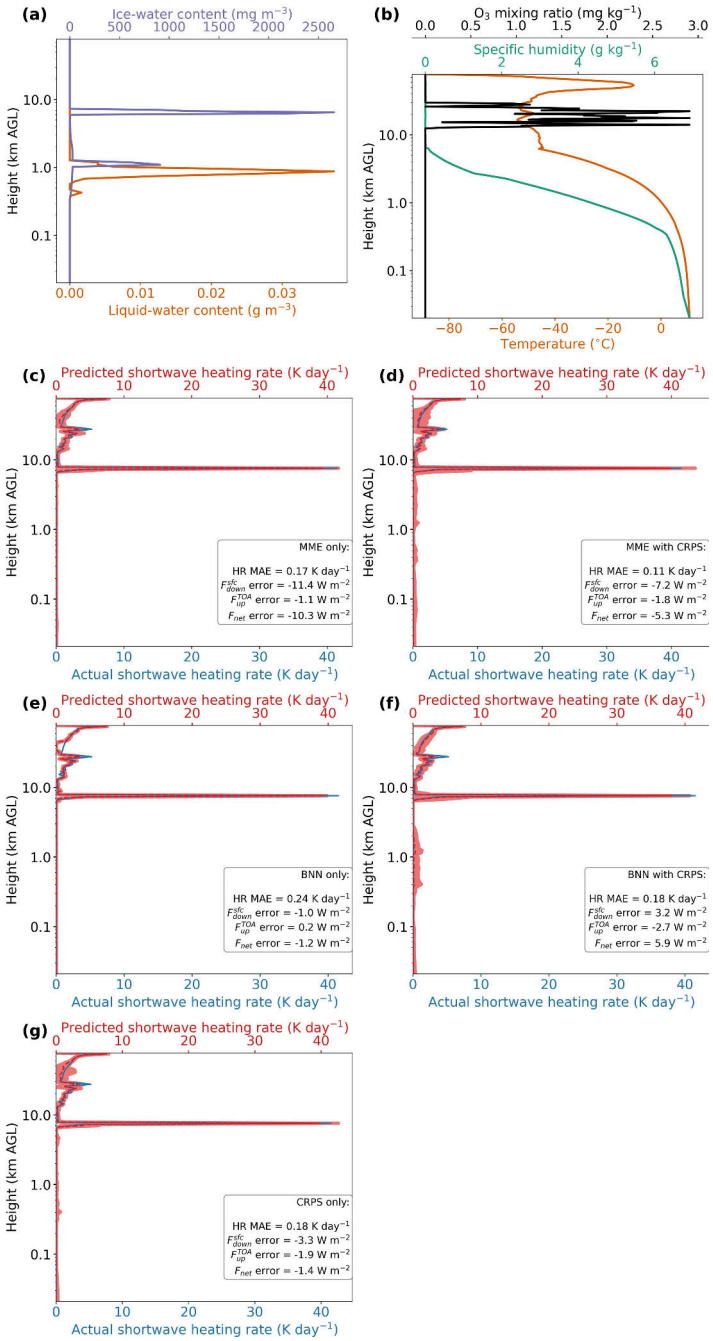


Figure 15: Case study in the testing data: 0000 UTC 29 Jan 2020, 41.65 °N, 168.52 °E. All formatting is explained in the caption of Figure 9.

7 Summary and future work

For a long time, uncertainty quantification (UQ) has been a key ambition for machine learning (ML) in environmental science (ES). The field of computer science has recently made breakthroughs in ML-UQ, and environmental scientists are just beginning to apply the resulting methods. One hope of the ES community is that new ML-UQ methods can be used to assess the trustworthiness of an ML model on a case-by-case basis – *e.g.*, to alert users when the model is expected to have a large error. However, ML-UQ methods, just like the base ML models with which they are coupled, do not generalize well to out-of-sample data. When a UQ-enhanced ML model encounters out-of-sample data, it is likely to produce a *catastrophic error* – *i.e.*, an extremely wrong prediction with high confidence. While scientists are generally aware that ML generalizes poorly out of sample, in our experience they do not have this awareness for UQ, leaving them prone to catastrophic errors. We wish to discourage overreliance on ML-UQ by showing that there are fundamentally unresolvable types of uncertainty, including that which arises from out-of-sample data.

To this end, we trained neural networks (NN) to predict shortwave radiative transfer. The NNs predict 130 quantities – a length-127 vector of heating rates and 3 flux components – each with a 50-member ensemble. The ensemble is produced by one of five UQ methods: a multi-model ensemble (MME), Bayesian neural network (BNN), training with the continuous ranked probability score (CRPS) loss function, or a hybrid method (MME/CRPS or BNN/CRPS). The validation and testing data are pushed out of sample by perturbing several predictor variables: temperature, humidity, liquid cloud, ice cloud, and ozone.

In Experiment 1, the NNs are trained with clean (unperturbed) data, then tasked with generalizing to *moderately perturbed* validation data and *heavily perturbed* testing data. Irrespective of the UQ method with which they are coupled, the NNs completely fail on the validation and testing data, generating poor point predictions (ensemble means) and uncertainty estimates. Even the best-performing UQ method (BNN/CRPS) is extremely overconfident for heating rates, producing only 14% as much spread as it should. Perturbations made to the ozone layer – which are more severe than perturbations made to other atmospheric properties – confound our NNs the most. The models fail on validation and testing data because the perturbations therein are simply not "seen" in the clean training data. In other words, the perturbations occur along basis vectors with little to no variability in the training data. While it is generally recognized that ML-based predictions will fail in this setting, ML-generated uncertainty estimates fail just as spectacularly. This has serious implications for operational use of ML, *e.g.*, in weather-forecasting. If a high-impact event occurs in real time that is not represented in the training data (*i.e.*, is out of sample), overreliance on ML – including ML-based uncertainty estimates – could have severe consequences.

The discouraging results from Experiment 1 motivated another question: what happens if the basis vectors represented by the perturbations are represented *just a little* in the training data? To answer this, in Experiment 2 we trained NNs with *lightly perturbed* data, calling these "LP-trained models" (as opposed to the clean-trained models in Experiment 1). On the validation and testing data, the LP-trained models performed much better than clean-trained models. This result illustrates the power of triggering each basis vector of variability – even just a little – in the training data. Ebert-Uphoff and Deng (2017) found a similar result for causal discovery in ES: if a causal mechanism is not triggered in the training data, it will not be learned by the model.

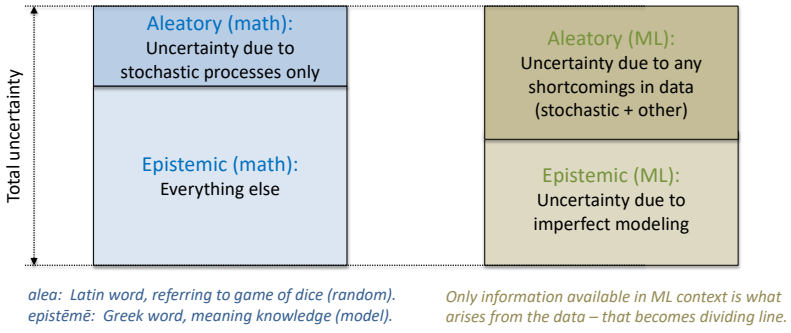
Despite the obvious advantages of the LP-trained models, evaluation on the testing data revealed some concerning properties. For example, the best-performing UQ method (BNN/CRPS) is still quite overconfident for heating rates, producing only 52% as much spread as it should. Thus, lightly triggering important basis vectors in the training data

allowed the ML-UQ models to extrapolate much better along these basis vectors, but it did not cure all ills. This is especially true for the testing data; the three concerns listed above are quite minor in the validation data. The above results have two important implications for operational ML. First, while enhancing the training data by triggering important basis vectors of the predictor space allows ML to better extrapolate along these vectors, ML will likely struggle when extrapolating *far* out of sample. Second, in large predictor spaces (which are common in ES), it is hard to know *all* the important basis vectors, especially those representing high-impact events. Thus, even for ML models with safeguards against poor out-of-sample performance – such as UQ or enhanced training data – we still discourage overreliance on ML and ML-UQ. Also, we encourage users to be familiar with the training data used for an ML model, so that they can identify out-of-sample (or poorly sampled) situations and approach the model with a healthy skepticism.

Future work will proceed along two lines. First, we will explore strategies for adapting ML-UQ to more realistic out-of-sample data, such as those caused by climate change (our application was a sandbox for testing the generalization of ML-UQ methods under extreme conditions). Second, we will try combining ML-UQ with tools that automatically detect out-of-sample data (Bulusu et al., 2020). Although these tools cannot improve an ML model's generalization to out-of-sample data, they can alert users when out-of-sample data appear. This would automate part of the process of determining an ML model's trustworthiness.

Appendix A Aleatory vs. epistemic uncertainty

Uncertainty can be divided into two components: aleatory and epistemic. Briefly, according to the ML literature, aleatory uncertainty is due to gaps in the (training) dataset, while epistemic uncertainty is due to gaps in model development. Note, however, that the definitions vary across disciplines – see Figure A1 and discussions in Hüllermeier and Waegeman (2021), Bevan (2022), Haynes et al. (2023)). We use the ML definition throughout this manuscript, shown in Figure A1b. Figure A1 is adapted from Figure 3 of Haynes et al. (2023).



(a) Math definition (b) ML definition

Figure A1: The aleatory/epistemic divide, according to different disciplines. [a] The math (original) definition is based only on the mathematical nature of the observed system. Only uncertainty due to stochastic processes, such as the chaotic nature of the atmosphere, is considered aleatory. [b] The ML definition of aleatory uncertainty is much wider, including not only uncertainty due to the stochastic nature of the system, but due to all other shortcomings of the dataset, such as limited observations.

In Section 1.3 we provide examples of unresolvable uncertainty, which cannot be captured by *any* ML-UQ method. One might wonder whether the unresolvable uncertainty in these scenarios is aleatory or epistemic. The answer is: it depends on how the dataset is chosen. We illustrate this below for Scenario 1 from Section 1.3, where uncertainty depends strongly on a variable x_{unknown} not included in the ML model. The key question in distinguishing aleatory from epistemic is: where was the information lost? Let us track our steps:

1. The physical system contains both variables: x_{known} and x_{unknown} (Equations 1).
2. The dataset collected by observing the physical system may or may not include x_{unknown}. Letting M be the number of samples, the two possibilities for the dataset are

$$\mathcal{D}_1 = \left\{(x^i_{\text{known}}, x^i_{\text{unknown}}); \qquad i = 1, 2, \ldots, M\right\} \text{ or}$$
$$\mathcal{D}_2 = \left\{(x^i_{\text{known}}); \qquad i = 1, 2, \ldots, M\right\}.$$

3. Regardless of which dataset was chosen ($\mathcal{D}_1$ or $\mathcal{D}_2$), the ML model has no access to x_{unknown} and depends only on x_{known}.

In other words, regardless of which dataset was chosen, the ML model is exactly the same. However, uncertainty in the model's output arising from its ignorance of x_{unknown} is considered **epistemic** if the dataset chosen is $\mathcal{D}_1$ (because the problem is deemed to be in the model), versus **aleatory** if the dataset chosen is $\mathcal{D}_2$ (because the problem is deemed to be in the data). In other words, the distinction of aleatory vs. epistemic depends on whether the relevant information was dropped during the data-collection or model-development step. The key conclusion for this study is that the types of unresolvable uncertainty in Section 1.3 can show up in both the aleatory and epistemic components; thus, we need to employ UQ methods that can capture both types.

Appendix B Open research

We used version 3.0.0 of ML4RT (Machine Learning for Radiative Transfer; `https://doi.org/10.5281/zenodo.10086129`) – a Python library managed by author Lagerquist – for all tasks in this study. The input data and all models not involved in a MME can be found at `https://zenodo.org/doi/10.5281/zenodo.10081204`; the MMEs can be found at `https://zenodo.org/doi/10.5281/zenodo.10084393`, `https://zenodo.org/doi/10.5281/zenodo.10084403`, `https://zenodo.org/doi/10.5281/zenodo.10084445`, and `https://zenodo.org/doi/10.5281/zenodo.10084454`.

Acknowledgments
This work was partially supported by the NOAA Global Systems Laboratory, Cooperative Institute for Research in the Atmosphere, and NOAA Award Number NA19OAR4320073. Author Ebert-Uphoff's work was partially supported by NSF AI Institute grant #2019758.

References

Baran, S., & Baran, A. (2021). Calibration of wind speed ensemble forecasts for power generation. *arXiv e-prints*, *2104*(14910). Retrieved from `https://arxiv.org/abs/2104.14910`

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

Beucler, T., Pritchard, M., Yuval, J., Gupta, A., Peng, L., Rasp, S., ... Gentine, P. (2021). Climate-invariant machine learning. *arXiv e-prints*, *2112*(08440). Retrieved from `https://arxiv.org/abs/2112.08440`

Bevan, L. (2022). The ambiguities of uncertainty: A review of uncertainty frameworks relevant to the assessment of environmental change. *Futures*, *137*, 102919. Retrieved from https://doi.org/10.1016/j.futures.2022.102919

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

Buiten, M. (2019). Towards intelligent regulation of artificial intelligence. *European Journal of Risk Regulation*, *10*(1), 41-59. Retrieved from https://doi.org/10.1017/err.2019.8

Bulusu, S., Kailkhura, B., Li, B., Varshney, P., & Song, D. (2020). Anomalous example detection in deep learning: A survey. *IEEE Access*, *8*, 132330-132347. Retrieved from https://doi.org/10.1109/ACCESS.2020.3010274

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

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

Cooney, J., Bowman, K., Homeyer, C., & Fenske, T. (2018). Ten year analysis of tropopause-overshooting convection using GridRad data. *Journal of Geophysical Research: Atmospheres*, *123*(1), 329-343. Retrieved from https://doi.org/10.1002/2017JD027718

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

Ebert-Uphoff, I., & Deng, Y. (2017). Causal discovery in the geosciences—Using synthetic data to learn how to interpret results. *Computers and Geosciences*, *99*, 50-60. Retrieved from https://doi.org/10.1016/j.cageo.2016.10.008

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

Ghazvinian, M., Zhang, Y., Seo, D., He, M., & Fernando, N. (2021). A novel hybrid artificial neural network-Parametric scheme for postprocessing medium-range precipitation forecasts. *Advances in Water Resources*, *151*, 103907. Retrieved from https://doi.org/10.1016/j.advwatres.2021.103907

Gil, Y., Pierce, S., Babaie, H., Banerjee, A., Borne, K., Bust, G., ... Shekhar, S. (2019). Intelligent systems for geosciences: An essential research agenda. *Communications of the Association for Computing Machinery*, *62*(1), 76-84. Retrieved from https://dl.acm.org/doi/10.1145/3192335

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press. Retrieved from https://www.deeplearningbook.org

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

Haynes, K., Lagerquist, R., McGraw, M., Musgrave, K., & Ebert-Uphoff, I. (2023). Creating and evaluating uncertainty estimates with neural networks for environmental-science applications. *Artificial Intelligence for the Earth Systems*, *2*(2), 1-29. Retrieved from https://doi.org/10.1175/AIES-D-22-0061.1

Hertel, V., Chow, C., Wani, O., Wieland, M., & Martinis, S. (2023). Probabilistic

SAR-based water segmentation with adapted Bayesian convolutional neural network. *Remote Sensing of Environment*, *285*, 113388. Retrieved from https://doi.org/10.1016/j.rse.2022.113388

Hoffman, M., Blei, D., Wang, C., & Paisley, J. (2013). Stochastic variational inference. *Journal of Machine Learning Research*, *14*(1), 1303-1347. Retrieved from https://www.jmlr.org/papers/volume14/hoffman13a/hoffman13a.pdf

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

Hüllermeier, E., & Waegeman, W. (2021). Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods. *Machine Learning*, *110*(3), 457-506. Retrieved from https://doi.org/10.1007/s10994-021-05946-3

Iacono, M., Delamere, J., Mlawer, E., Shephard, M., Clough, S., & Collins, W. (2008). Radiative forcing by long-lived greenhouse gases: Calculations with the AER radiative transfer models. *Journal of Geophysical Research: Atmospheres*, *113*(D13). Retrieved from https://doi.org/10.1029/2008JD009944

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

Kim, P., & Song, H. (2022). Usefulness of automatic hyperparameter optimization in developing radiation emulator in a numerical weather prediction model. *Atmosphere*, *13*(5), 721. Retrieved from https://doi.org/10.3390/atmos13050721

Kingma, D., & Welling, M. (2013). Auto-encoding variational Bayes. *arXiv e-prints*, *1312*(6114). Retrieved from https://arxiv.org/abs/1312.6114

Klotz, D., Kratzert, F., Gauch, M., Sampson, A., Brandstetter, J., Klambauer, G., ... Nearing, G. (2022). Uncertainty estimation with deep learning for rainfall–runoff modeling. *Hydrology and Earth System Sciences*, *26*(6), 1673-1693. Retrieved from https://doi.org/10.5194/hess-26-1673-2022

Krasnopolsky, V., Belochitski, A., Hou, Y., Lord, S., & Yang, F. (2012). *Accurate and fast neural network emulations of long and short wave radiation for the NCEP Global Forecast System model* (Vol. Office Note 471; Tech. Rep.). Retrieved from https://repository.library.noaa.gov/view/noaa/6951

Lagerquist, R., Turner, D., Ebert-Uphoff, I., & Stewart, J. (2023). Estimating full longwave and shortwave radiative transfer with neural networks of varying complexity. *Journal of Atmospheric and Oceanic Technology*, *conditionally accepted*. Retrieved from https://doi.org/10.22541/essoar.168319865.58439449/v1

Lagerquist, R., Turner, D., Ebert-Uphoff, I., Stewart, J., & Hagerty, V. (2021). Using deep learning to emulate and accelerate a radiative transfer model. *Journal of Atmospheric and Oceanic Technology*, *38*(10), 1673-1696. Retrieved from https://doi.org/10.1175/JTECH-D-21-0007.1

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

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

Ranganath, R., Gerrish, S., & Blei, D. (2014). Black box variational inference. *Proceedings of Machine Learning Research*, *33*, 814-822. Retrieved from http://

proceedings.mlr.press/v33/ranganath14

Rasp, S., Pritchard, M., & Gentine, P. (2018). Deep learning to represent subgrid processes in climate models. *Proceedings of the National Academy of Sciences*, *115*(39), 9684-9689. Retrieved from https://doi.org/10.1073/pnas.1810286115

Reichstein, M., Camps-Valls, G., Stevens, B., Jung, M., Denzler, J., Carvalhais, N., & Prabhat. (2019). Deep learning and process understanding for data-driven Earth system science. *Nature*, *566*, 195-204. Retrieved from https://doi.org/10.1038/s41586-019-0912-1

Rezende, D., Mohamed, S., & Wierstra, D. (2014). Stochastic backpropagation and variational inference in deep latent Gaussian models. *International Conference on Machine Learning*, *2*, 2. Retrieved from http://web2.cs.columbia.edu/~blei/fogm/2018F/materials/RezendeMohamedWierstra2014.pdf

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *International conference on medical image computing and computer-assisted intervention.* Munich, Germany. Retrieved from https://doi.org/10.1007/978-3-319-24574-4_28

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

Scheuerer, M., Switanek, M., Worsnop, R., & Hamill, T. (2020). Using artificial neural networks for generating probabilistic subseasonal precipitation forecasts over California. *Monthly Weather Review*, *148*(8), 3489-3506. Retrieved from https://doi.org/10.1175/MWR-D-20-0096.1

Schulz, B., & Lerch, S. (2022). Machine learning methods for postprocessing ensemble forecasts of wind gusts: A systematic comparison. *Monthly Weather Review*, *150*(1), 235-257. Retrieved from https://doi.org/10.1175/MWR-D-21-0150.1

Slovnik, W. (1992). *International Meteorological Vocabulary* (Tech. Rep.). World Meteorological Organization.

Song, H., & Roh, S. (2021). Improved weather forecasting using neural network emulation for radiation parameterization. *Journal of Advances in Modeling Earth Systems*, *13*(10). Retrieved from https://doi.org/10.1029/2021MS002609

Van, T., Nguyen, T., Tran, N., Nguyen, H., Doan, L., Dao, H., & Minh, T. (2020). Interpreting the latent space of generative adversarial networks using supervised learning. *International Conference on Advanced Computing and Applications*, 49-54. Retrieved from https://doi.org/10.1109/ACOMP50827.2020.00015

Veldkamp, S., Whan, K., Dirksen, S., & Schmeits, S. (2021). Statistical postprocessing of wind speed forecasts using convolutional neural networks. *Monthly Weather Review*, *149*(4), 1141-1152. Retrieved from https://doi.org/10.1175/MWR-D-20-0219.1

Wallace, J., & Hobbs, P. (2006). *Atmospheric Science: An Introductory Survey* (Vol. 2). Elsevier.

Wen, Y., Vicol, P., Ba, J., Tran, D., & Grosse, R. (2018). Flipout: Efficient pseudo-independent weight perturbations on mini-batches. *International Conference on Learning Representations*. Retrieved from https://arxiv.org/pdf/1803.04386.pdf

Wimmers, A., Velden, C., & Cossuth, J. (2019). Using deep learning to estimate tropical cyclone intensity from satellite passive microwave imagery. *Monthly Weather Review*, *147*(6), 2261-2282. Retrieved from https://doi.org/10.1175/MWR-D-18-0391.1

Zhou, Z., Siddiquee, M., Tajbakhsh, N., & Liang, J. (2019). Unet++: Redesigning skip connections to exploit multiscale features in image segmentation. *IEEE Transactions on Medical Imaging*, *39*(6), 1856-1867. Retrieved from https://

doi.org/10.1109/TMI.2019.2959609

Machine-learned uncertainty quantification is not magic: Lessons learned from emulating radiative transfer with ML
Supplemental material

Ryan Lagerquist[1,2*], Imme Ebert-Uphoff[1,3], David D. Turner[2], and Jebb Q. Stewart[2]

[1]Cooperative Institute for Research in the Atmosphere (CIRA), Colorado State University, Fort Collins, Colorado
[2]National Oceanic and Atmospheric Administration (NOAA) Global Systems Laboratory (GSL), Boulder, Colorado
[3]Department of Electrical and Computer Engineering, Colorado State University, Fort Collins, Colorado

1 Methods for data perturbation

This section explains how each atmospheric property is perturbed, with the exception of near-surface temperature (see Section 2d of main text).

1.1 Near-surface humidity

Our motivation is to mimic the lower-tropospheric moistening expected with climate change. The procedure has three parameters: maximum depth of the moist layer ($D_{\max}$), minimum surface relative humidity ($\text{RH}_{\text{sfc}}^{\min}$), and maximum ($\text{RH}_{\text{sfc}}^{\max}$). Parameter settings are shown in Table S1; the procedure is shown schematically in Figure S1. After the numbered procedure below, we recompute the two moisture variables used as predictors (relative and specific humidity), based on the new mixing ratio and untouched temperature/pressure.

1. Sample to determine the depth of the moist layer: $D \in \mathcal{U}[0, D_{\max}]$.
2. Sample to determine the surface RH: $\text{RH}_{\text{sfc}} \in \mathcal{U}\left[\text{RH}_{\text{sfc}}^{\min}, \text{RH}_{\text{sfc}}^{\max}\right]$.
3. Compare the new and original (unperturbed) surface-RH values. If $\text{RH}_{\text{sfc}} \leq \text{RH}_{\text{sfc}}^{\text{orig}}$, do nothing and end the procedure.
4. Convert surface RH to surface mixing ratio, w_{sfc}.
5. Calculate the increase in surface mixing ratio: $\Delta w_{\text{sfc}} = w_{\text{sfc}} - w_{\text{sfc}}^{\text{orig}}$.
6. At each height in the moist layer, scale the mixing-ratio increase linearly from Δw_{sfc} at the surface to 0 at height D above the surface. See Figures S1a-c.
7. If step 6 led to any height with dewpoint > temperature, reduce dewpoint to temperature. See Figure S1d.
8. If step 6 led to any mixing ratio above 40 g kg^{-1}, reduce to 40 g kg^{-1}. See Figure S1e.

*325 Broadway, R/GSL6, Boulder, CO 80305

Corresponding author: Ryan Lagerquist, ralager@colostate.edu

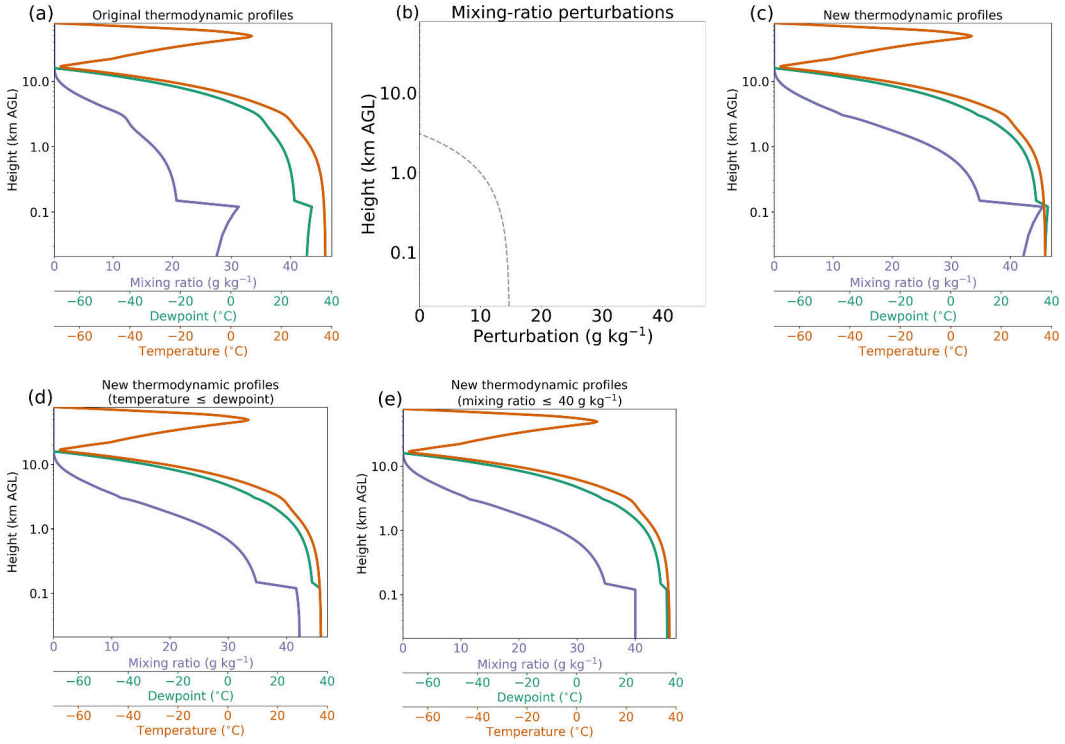


Figure S1: Procedure for perturbing near-surface humidity. In this example, the moist-layer depth D is 3 km and the new surface relative humidity $\mathrm{RH_{sfc}}$ is 100%. In panel c, the new mixing ratio is obtained by adding panels a and b; the new dewpoint is then computed from the new mixing ratio. In panel d, the new dewpoint is obtained by taking the minimum of dewpoint and temperature at each height; the new mixing ratio is then computed from the new dewpoint. In panel e, the new mixing ratio is obtained by reducing to 40 g $\mathrm{kg^{-1}}$ at each height if necessary; the new dewpoint is then computed from the new mixing ratio.

Table S1: Parameter settings for perturbation of predictor variables.

Parameter	**Setting for lightly perturbed training data**	**Setting for validation data**	**Setting for testing data**
	Near-surface temperature		
$D_{\max}$	1.25 km	2.5 km	5 km
$\Delta T_{\text{sfc}}^{\max}$	2 K	4 K	8 K
	Near-surface humidity		
$D_{\max}$	1.25 km	2.5 km	5 km
$\text{RH}_{\text{sfc}}^{\min}$	50%	50%	50%
$\text{RH}_{\text{sfc}}^{\max}$	62.5%	75%	100%
	Liquid cloud		
$N_{\max}$	2	3	5
$D_{\max}$	5 km	5 km	5 km
$\text{LWC}_{\text{center}}^{\max}$	2 g m^{-3}	2.5 g m^{-3}	5 g m^{-3}
σ_{LWC}	0.25 g m^{-3}	0.5 g m^{-3}	1 g m^{-3}
	Ice cloud		
$N_{\max}$	2	3	5
$D_{\max}$	5 km	5 km	5 km
$\text{IWC}_{\text{center}}^{\max}$	2 g m^{-3}	2.5 g m^{-3}	5 g m^{-3}
σ_{IWC}	0.25 g m^{-3}	0.5 g m^{-3}	1 g m^{-3}
	Ozone		
$D_{\min}$ and $D_{\max}$	40 and 60 km	25 and 60 km	0.1 and 60 km
$z_{\text{center}}^{\min}$ and $z_{\text{center}}^{\max}$	20 and 50 km AGL	20 and 50 km AGL	20 and 50 km AGL
$\text{w}_{\text{center}}^{\max}$	20 mg kg^{-1}	25 mg kg^{-1}	50 mg kg^{-1}
σ_{w}	0.25 mg kg^{-1}	0.5 mg kg^{-1}	1 mg kg^{-1}

1.2 Liquid cloud

Our motivation is to create more complex cloud profiles, as well as denser and deeper clouds, than seen in the real atmosphere. The procedure has four parameters: maximum number of cloud layers ($N_{\max}$), maximum layer depth ($D_{\max}$), maximum liquid-water content at layer center ($\text{LWC}_{\text{center}}^{\max}$), and noise level for LWC (σ_{LWC}). Parameter settings are shown in Table S1; the procedure is shown schematically in Figure S2.

1. Sample to determine the number of cloud layers: $N \in \mathcal{U}[0, N_{\max}]$.
2. For each cloud layer i from $1 \ldots N$:
   (a) Sample to determine the depth of the i^{th} cloud: $D \in \mathcal{U}[0, D_{\max}]$.
   (b) Sample to determine the height of the cloud top above the surface:

$z_{\text{top}} \in \mathcal{U}\,[0, z_{\text{tropopause}} + 2\text{ km}]$.[2,3]

(c) Calculate the height of the cloud bottom: $z_{\text{bottom}} = z_{\text{top}} - D$. If this leads to $z_{\text{bottom}} < 0$ km AGL, increase to 0 km AGL. See Figure S2b.

(d) Find all grid points in the cloud. These are grid points with a height in $[z_{\text{bottom}}, z_{\text{top}}]$ and temperature $\geq$ -40 °C that have not already been assigned to another liquid-cloud layer.[4] See Figure S2c.

(e) Sample to determine the LWC at the center of the cloud: $\text{LWC}_{\text{center}} \in \mathcal{U}\,[0, \text{LWC}^{\text{max}}_{\text{center}}]$.

(f) At each height in the cloud, scale the LWC linearly from $\text{LWC}_{\text{center}}$ at the center to 0 g m^{-3} at both the top and bottom. See Figure S2d.

(g) Add Gaussian noise to the LWC profile for this cloud. Specifically, at each height in the cloud, add an offset δ, sampled from a normal distribution with mean = 0 g m^{-3} and standard deviation = σ_{LWC}. Symbolically, $\delta \in \mathcal{N}(0, \sigma_{\text{LWC}})$. See Figure S2e.

(h) If the previous step led to any LWC $<$ 0 g m^{-3}, increase to 0 g m^{-3}. If the previous step led to any LWC $> \text{LWC}_{\text{center}}$, reduce to $\text{LWC}_{\text{center}}$. See Figure S2f.

[2] We use the World Meteorological Organization (Slovnik, 1992) definition of the first tropopause: the lowest height at which lapse rate decreases to $<$ 2 K km^{-1} (let this be z'), provided that the mean lapse rate between z' and $z' + 2$ km does not exceed 2 K km^{-1}.

[3] A maximum height of $z_{\text{tropopause}} + 2$ km allows clouds to reach 2 km into the stratosphere, as in the overshooting tops of thunderstorms. Figure 12 of Cooney et al. (2018) shows that few overshooting tops reach further than 2 km into the stratosphere.

[4] Supercooled liquid droplets can exist at temperatures down to ∼-40 °C; see Figure 6.29 of Wallace and Hobbs (2006).

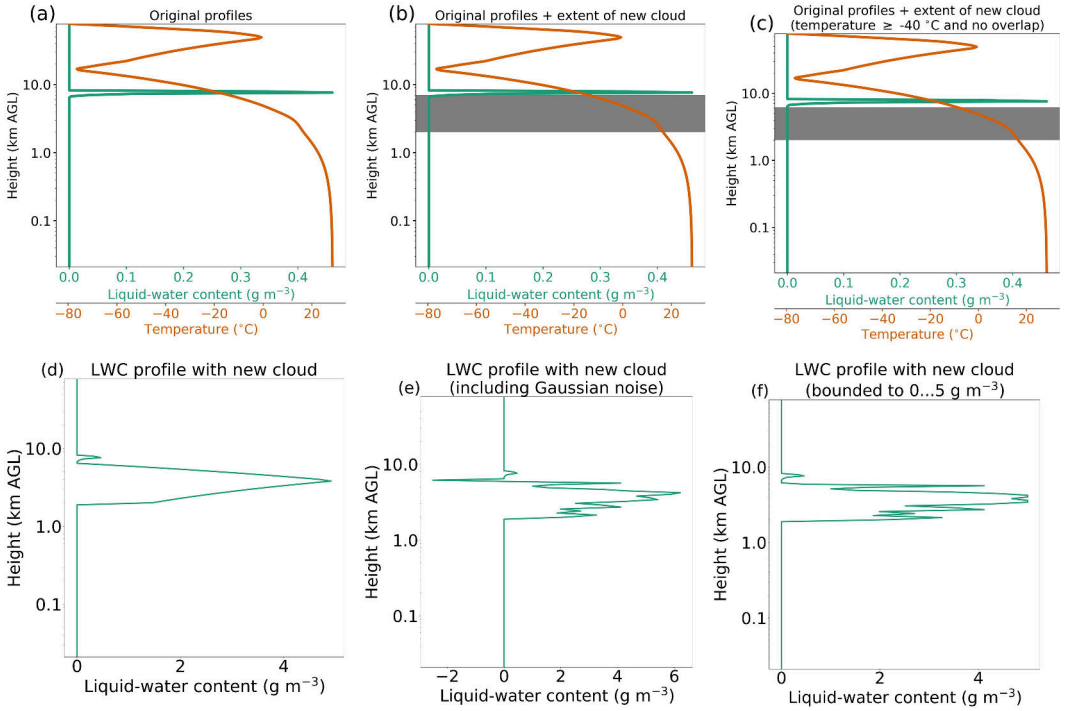


Figure S2: Procedure for creating a new liquid cloud layer. [a] Original profiles of LWC and temperature. Temperature is not perturbed in this procedure, but it is shown as a reference variable, because liquid droplets cannot exist at temperatures below -40 °C. [b] Same as panel a, but the extent of the proposed new cloud is shaded in grey. In this example, the proposed new cloud has depth $D = 5$ km. [c] Same as panel b, but corrected to exclude temperatures $< -40°$C and overlap with other liquid cloud. [d] LWC profile after adding new cloud. In this example, the LWC at the center of the cloud is $\text{LWC}_{\text{center}} = 5$ g m^{-3}. [e] Same as panel d, but after adding Gaussian noise for new cloud. In this example, the noise parameter is $\sigma_{\text{LWC}} = 1$ g m^{-3}. [f] Same as panel e, but after removing unwanted values created by Gaussian noise.

1.3 Ice cloud

This procedure has the same parameters as for liquid cloud, but replacing liquid-water content with ice-water content (IWC). In other words, replace $\text{LWC}_{\text{center}}^{\text{max}}$ with $\text{IWC}_{\text{center}}^{\text{max}}$ and σ_{LWC} with σ_{IWC}. Parameter settings are shown in Table S1.

The procedure itself – shown schematically in Figure S3 – is the same as for liquid cloud, except in step 2d. The criterion "temperature $\geq$ -40 °C" is replaced with "temperature $<$ 0 °C".

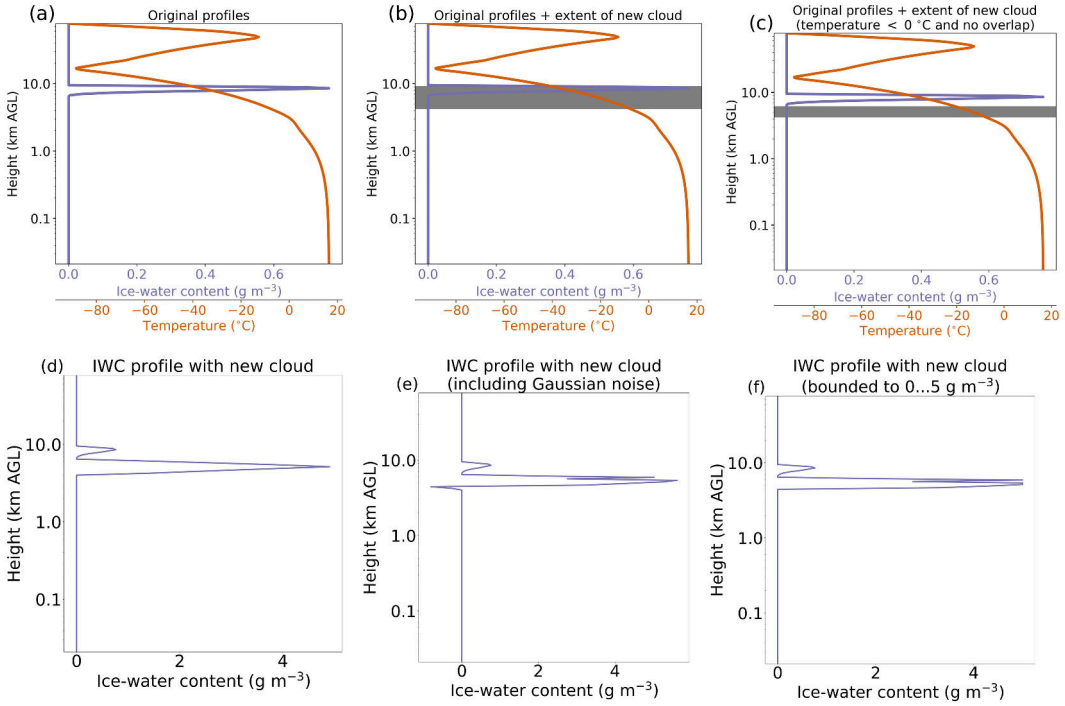


Figure S3: Procedure for creating a new ice cloud layer. [a] Original profiles of IWC and temperature. [b] Same as panel a, but the extent of the proposed new cloud is shaded in grey. In this example, the proposed new cloud has depth $D = 5$ km. [c] Same as panel b, but corrected to exclude temperatures $\geq 0°$C and overlap with other ice cloud. [d] IWC profile after adding new cloud. In this example, the IWC at the center of the cloud is $\text{IWC}_{\text{center}} = 5$ g m^{-3}. [e] Same as panel d, but after adding Gaussian noise for new cloud. In this example, the noise parameter is $\sigma_{\text{IWC}} = 1$ g m^{-3}. [f] Same as panel e, but after removing unwanted values created by Gaussian noise.

1.4 Ozone

Our motivation is to create more complex ozone layers – over a wider range of locations, depths, and mixing ratios – than seen in the real atmosphere. The procedure has six parameters: minimum and maximum depth (D_{min} and D_{max}), minimum and maximum height of layer center ($z_{\text{center}}^{\text{min}}$ and $z_{\text{center}}^{\text{max}}$), maximum ozone mixing ratio at layer center ($\text{w}_{\text{center}}^{\text{max}}$), and noise level for mixing ratio (σ_{w}). Parameter settings are shown in Table S1; the procedure is shown schematically in Figure S4.

1. Sample to determine the ozone-layer depth: $D \in \mathcal{U}\left[D_{\text{min}}, D_{\text{max}}\right]$.
2. Sample to determine the height of the layer center: $z_{\text{center}} \in \mathcal{U}\left[z_{\text{center}}^{\text{min}}, z_{\text{center}}^{\text{max}}\right]$.
3. Find all grid points in the ozone layer. These are grid points with a height in $\left[z_{\text{center}} - \frac{1}{2}D, z_{\text{center}} + \frac{1}{2}D\right]$ that are above the tropopause.
4. Sample to determine the ozone mixing ratio at the layer center: $w_{\text{center}} \in \mathcal{U}\left[0, w_{\text{center}}^{\text{max}}\right]$.
5. At each height in the ozone layer, scale the mixing ratio linearly from w_{center} at the center to 0 mg kg^{-1} at both the top and bottom.
6. Add Gaussian noise to the mixing-ratio profile. Specifically, at each height in the ozone layer, add an offset δ sampled from $\mathcal{N}(0, \sigma_w)$.
7. If the previous step led to any $w < 0$ mg kg^{-1}, increase to 0 mg kg^{-1}. If the previous step led to any $w > w_{\text{center}}$, reduce to w_{center}.

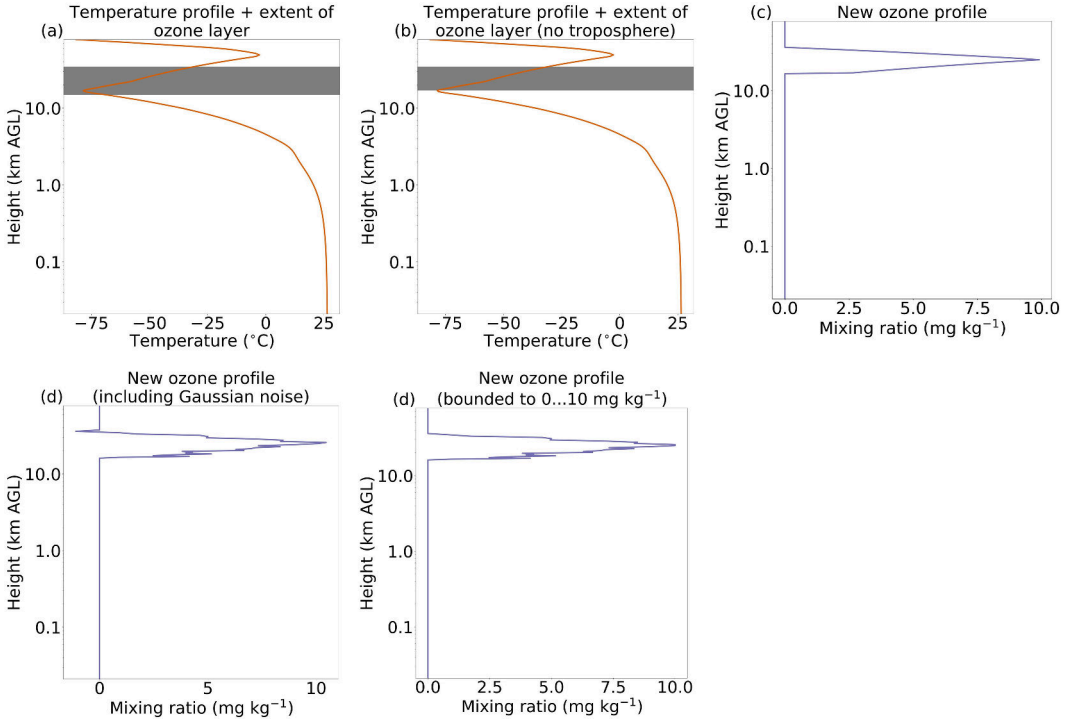


Figure S4: Procedure for creating a new ozone layer. [a] Original temperature profile, with proposed extent of ozone layer shaded in grey. In this example, the proposed ozone layer has depth $D = 20$ km and center $z_{\text{center}} = 25$ km AGL. The tropopause is at 16.5 km AGL, where temperature begins to increase with height. [b] Same as panel a, but corrected to exclude heights below the tropopause. [c] New ozone profile. In this example, $w_{\text{center}} = 10$ mg kg^{-1}. [d] Same as panel c, but after adding Gaussian noise. In this example, the noise parameter is $\sigma_w = 1$ mg kg^{-1}. [e] Same as panel d, but after removing unwanted values created by Gaussian noise.

2 Effects of data perturbation

Figures S5-S9 show the effects of different levels of data perturbation: light (for one set of training data), moderate (for the validation data), and heavy (for the testing data). Specific perturbation methods are discussed in Section 3d of the main text and Supplemental Section 1. A key property shown in these figures is that the perturbations to ozone are more drastic than those to liquid and ice water, which in turn are much more drastic than the perturbations to temperature and humidity.

Note that, as in Table S1, temperature is perturbed only at heights below $\{1.25, 2.5, 5\}$ km AGL in the {lightly perturbed training, validation, testing} data. Thus, above 5 km, the temperature distribution is nearly identical across the four datasets (Figure S5). All differences above 5 km are caused by differences among the *clean* datasets, *i.e.*, before perturbation. The same is true for other quantities not affected by perturbation: specific humidity above 5 km, liquid-water content in the stratosphere, ice-water content in the stratosphere, and ozone mixing ratio in the troposphere.

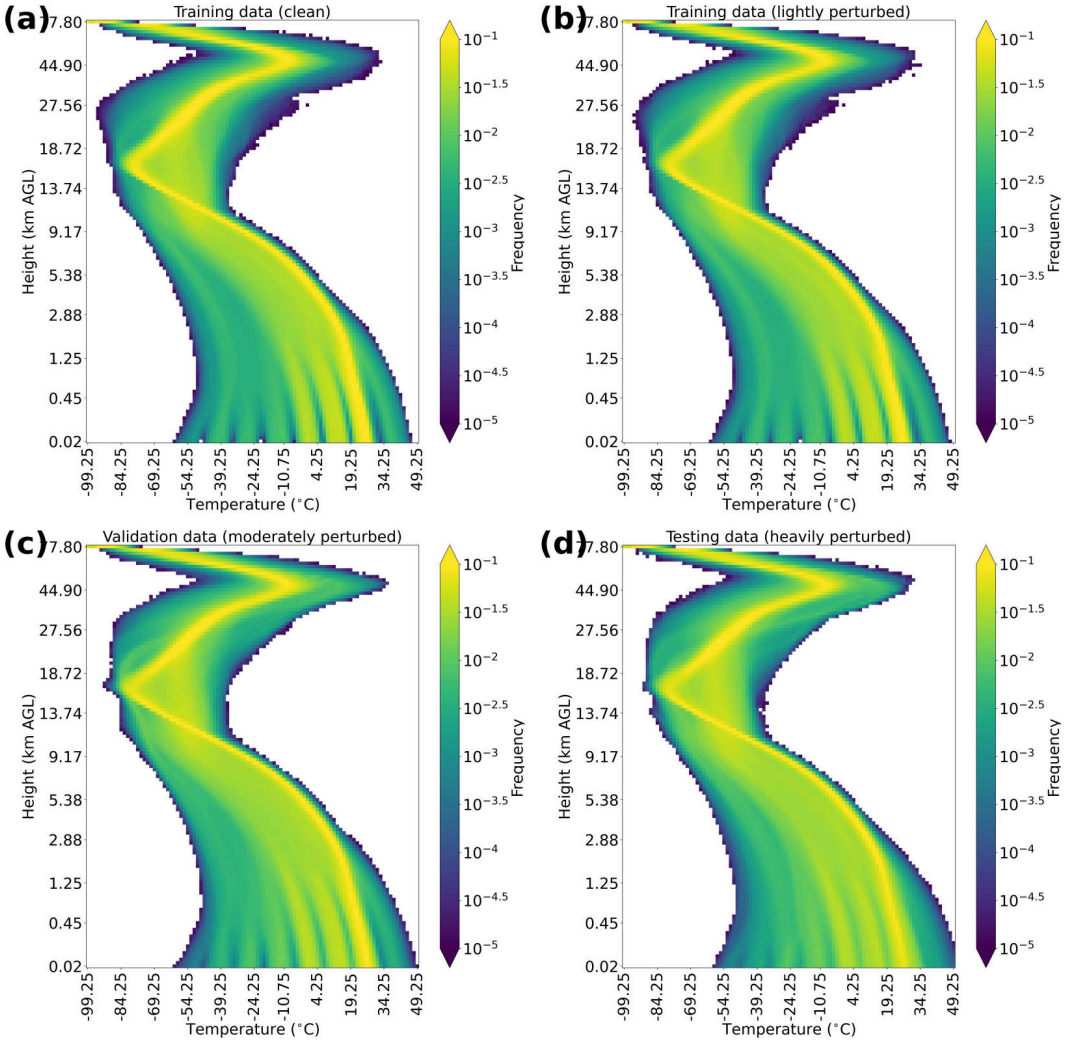


Figure S5: Distribution of temperature in [a] the clean training data, [b] the lightly perturbed training data, [c] the validation data, and [d] the testing data.

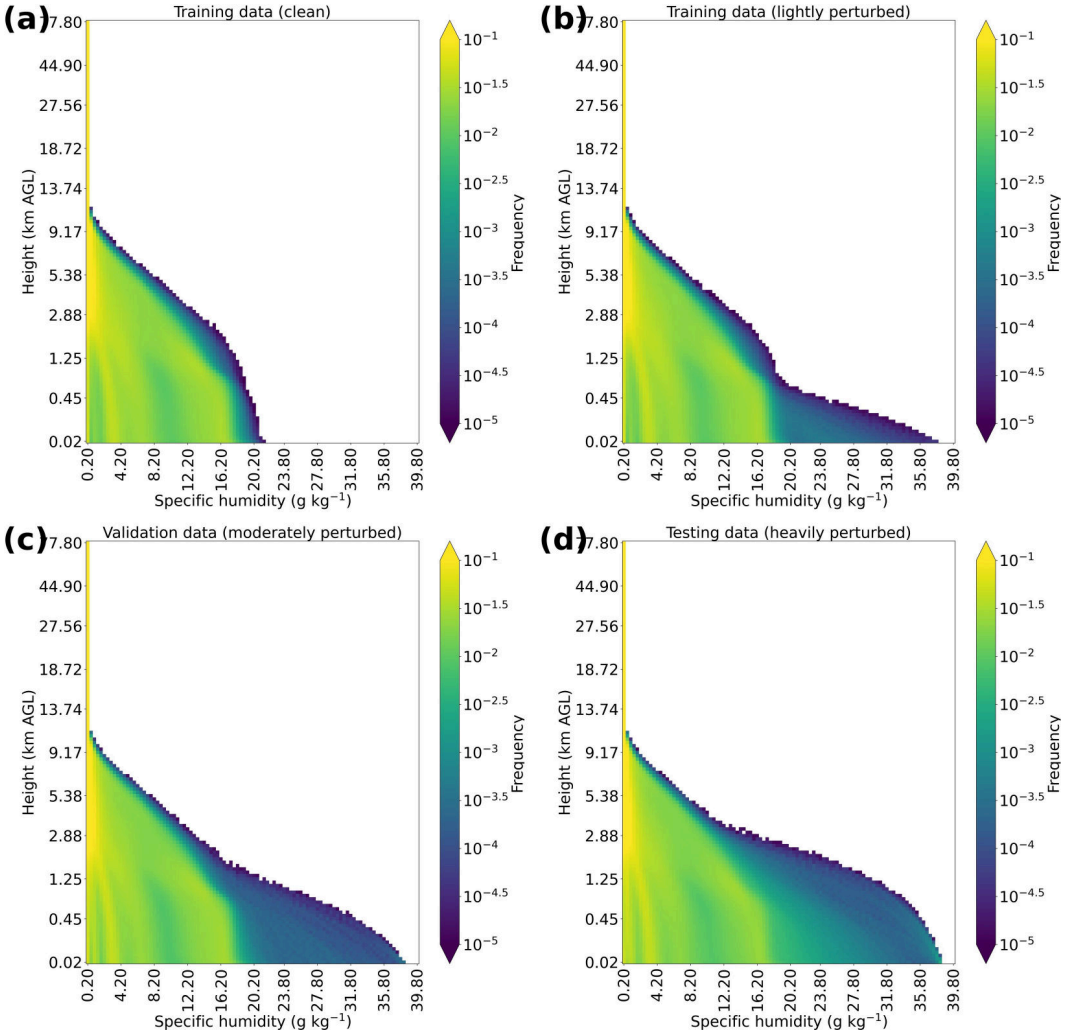


Figure S6: Same as Figure S5 but for specific humidity.

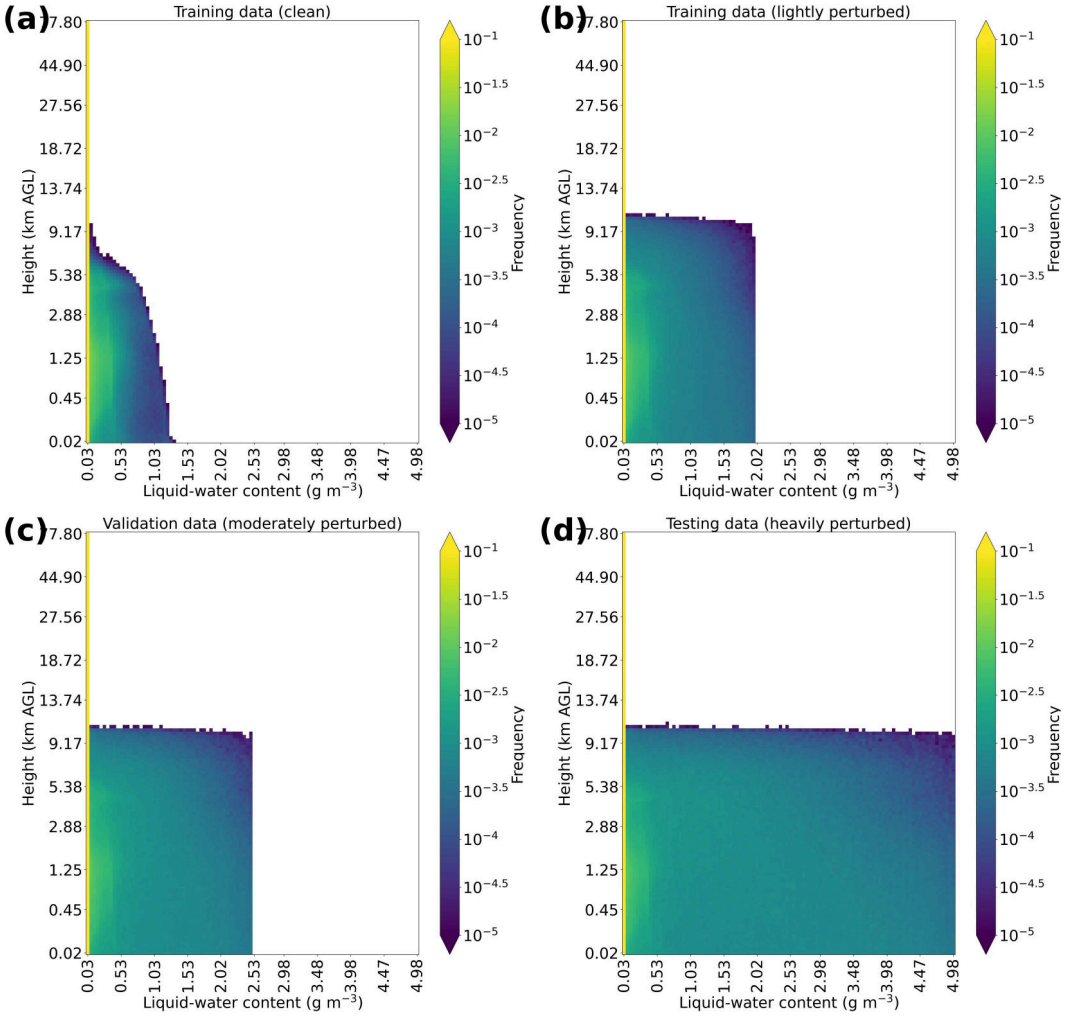


Figure S7: Same as Figure S5 but for liquid-water content (LWC).

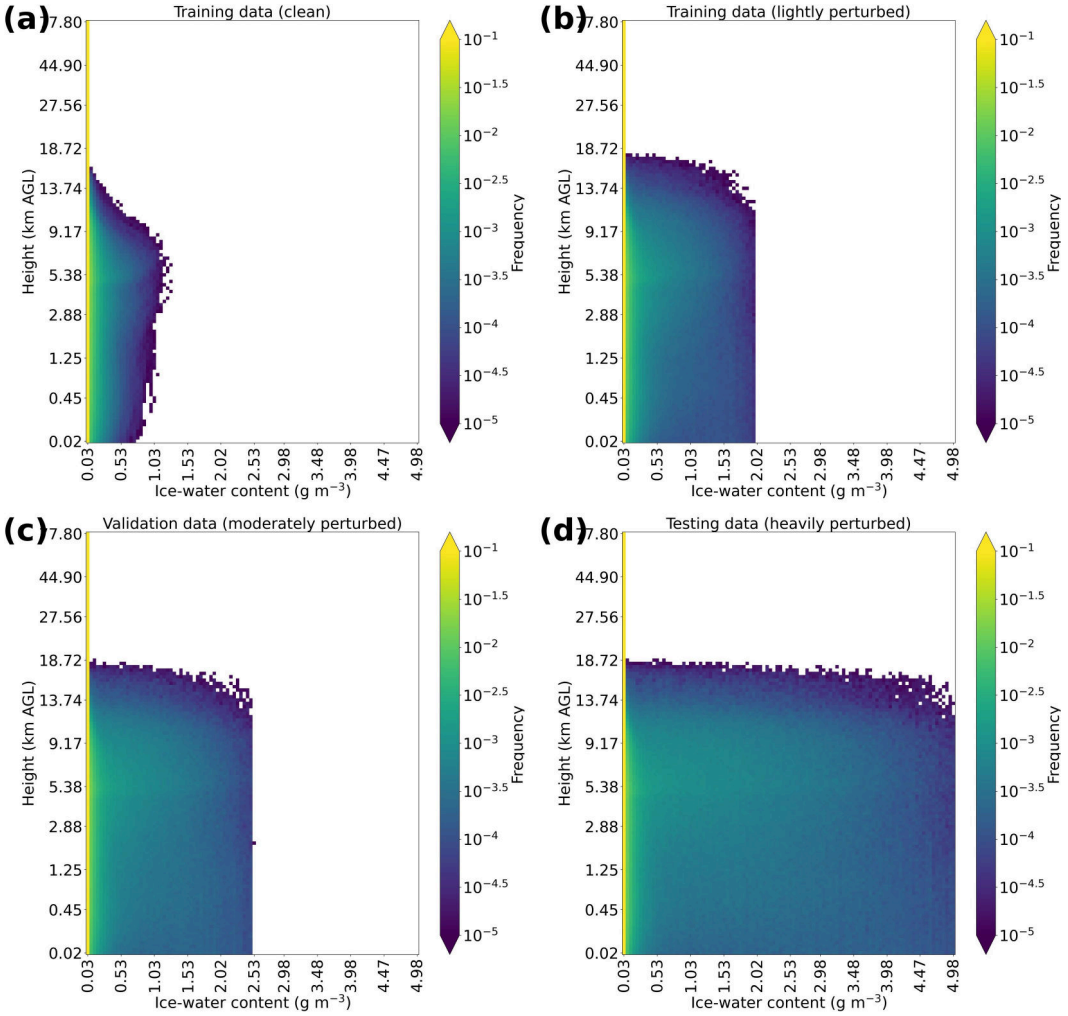


Figure S8: Same as Figure S5 but for ice-water content (IWC).

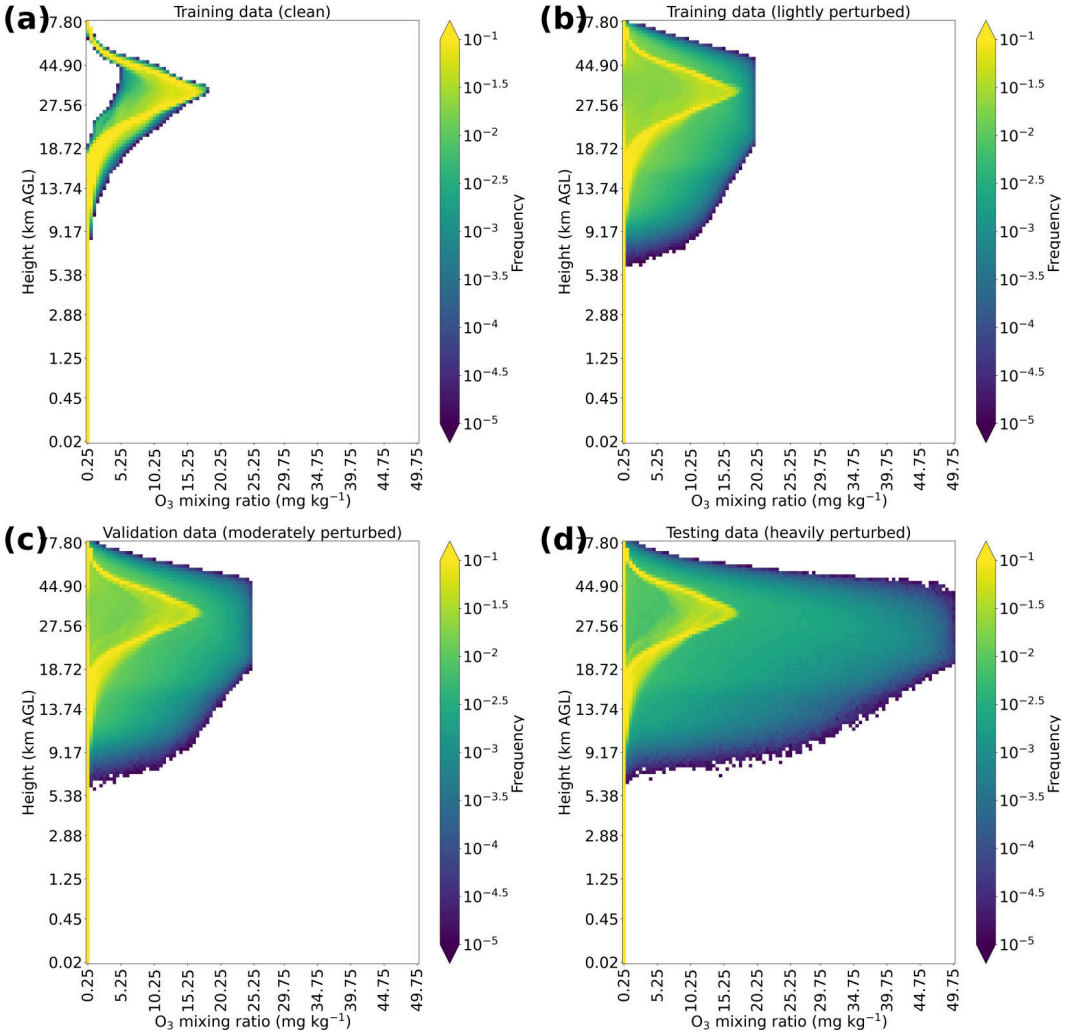


Figure S9: Same as Figure S5 but for ozone mixing ratio.

3 Hyperparameter experiment to determine the best BNN architecture

We conduct four hyperparameter experiments, each to determine the best BNN (Bayesian neural network) architecture in a given context. The contexts are:

- for the BNN-only UQ method, trained with clean data;
- for the BNN/CRPS method, trained with clean data;
- for the BNN-only method, trained with lightly perturbed data;
- for the BNN/CRPS method, trained with lightly perturbed data.

For each context we optimize four hyperparameters: the number of Bayesian fully connected layers (N_b^{fully}), number of Bayesian upsampling connections ($N_b^{\text{upsampling}}$), number of Bayesian skip connections (N_b^{skip}), and training method for Bayesian layers (reparameterization or flipout). Bayesian fully connected layers allow the network to do UQ for scalar outputs (fluxes), while Bayesian upsampling and skip connections allow the network to do UQ for vector outputs (heating rates). The attempted values for each hyperparameter are listed in Table S2.

Table S2: Hyperparameters optimized for BNNs.

Hyperparameter	**Values attempted**
Number of Bayesian fully connected layers	2, 3, 4
Number of Bayesian upsampling connections	1, 2
Number of Bayesian skip connections	1, 2
Training method for Bayesian layers	Reparameterization, flipout
Spectral complexity	64, 128

We experiment with the first four hyperparameters in particular – defined in Section 3a of the main text – because they are the main choices involved in changing a U-net from deterministic to Bayesian. When making fully connected layers Bayesian, we start at the layer nearest to the scalar outputs and work backwards. For example, if $N_b^{\text{fully}} = 3$, we make the two output layers (labeled "A" in Figure S10) and the preceding layer ("B" in Figure S10) Bayesian. Similarly, when making upsampling and skip connections Bayesian, we start at the layer nearest to the vector outputs and work backwards. For example, if $N_b^{\text{upsampling}} = 2$ and $N_b^{\text{skip}} = 1$, the two upsampling connections with highest spatial resolution ("C" in Figure S10) and one skip connection with highest spatial resolution ("D" in Figure S10) are made Bayesian. The vector output layer ("E" in Figure S10) is always Bayesian if the network is Bayesian. Within a network, the training method (reparameterization or flipout) is the same for all Bayesian layers.

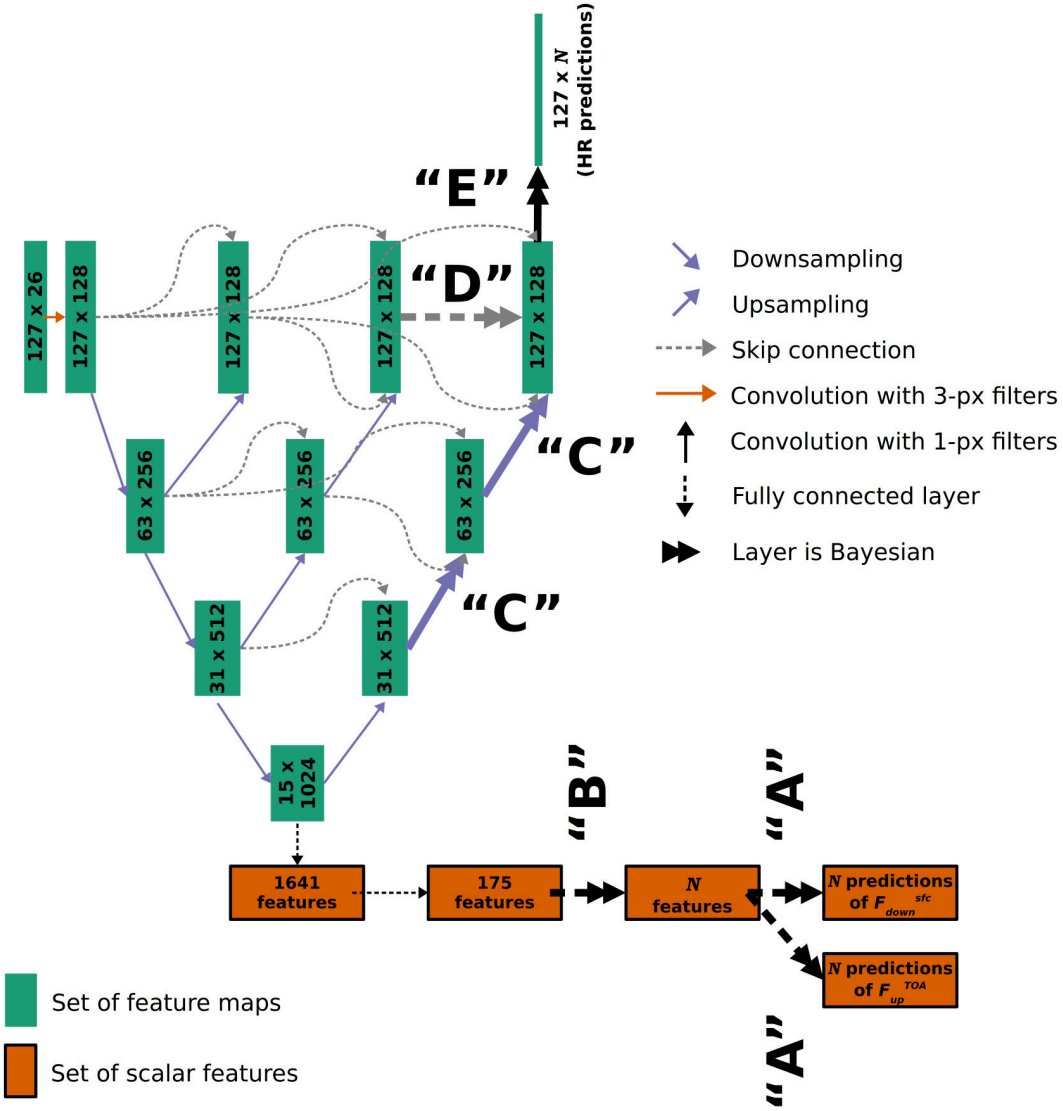


Figure S10: The optimal U-net++ setup for one context: the BNN-only UQ method with lightly perturbed training data. Since BNN-only uses the deterministic loss function rather than the CRPS, N (the ensemble size) = 1. This figure, which shows one specific U-net++ setup, is analogous to Figure 4 in the main text, which shows the generic U-net++ setup for UQ. In Figure 4 the double arrows indicate a component that *might be* Bayesian, while in this figure the double arrows indicate a component that *is* Bayesian.

For all hyperparameters not related to UQ, we use the optimal value determined in Lagerquist et al. (2023, henceforth L23), with one exception. The exception is spectral complexity, defined as the number of filters in the first convolutional layer (128 in Figure S10, the optimal value determined by L23). With the BNN/CRPS UQ method, this spectral complexity makes the network so large that training leads to out-of-memory errors. Thus, we attempt values of 64 and 128, as L23 found that a spectral complexity of 64 is nearly optimal.

The hyperparameter experiment is a grid search (Section 11.4.3 of Goodfellow et al., 2016), meaning that we try all 48 possible combinations of the hyperparameters. We use the validation data to find the best model, *i.e.*, best combination of hyperparame-

ter values. Because we wish to optimize both point predictions (ensemble means) and uncertainty estimates for two variables (heating rates [HR] and fluxes), choosing the "best" model is non-trivial, as model performance can be described by a wide variety of error metrics. We examine the 12 metrics shown in Figure 5 of the main text: {MAE, REL, SSREL, SSRAT, PITD, and MF} for {HR, flux}. MAE is mean absolute error; REL is reliability; SSREL is spread-skill reliability; SSRAT is spread-skill ratio; PITD is deviation from the perfect probability integral transform (PIT) histogram; and MF is monotonicity fraction measured from the discard test. MAE and REL concern point predictions only, while the other metrics concern the entire predicted distribution, including uncertainty. MAE, REL, SSREL, and PITD are negatively oriented with a perfect value of 0.0; MF is positively oriented with a perfect value of 1.0; and SSRAT has a perfect value of 1.0, with values of $[0, \infty)$ possible. We choose the best model by subjectively combining results across the 12 metrics.

For brevity – and because results are similar across the four contexts – here we show the results for only one context, namely the BNN-only method trained with lightly perturbed data. These results are shown in Figures S11-S22. In each figure, the circle represents the selected model (based on all 12 metrics) and the star represents the model with the best value of the given metric. Note that the best values for HR MF (Figure S16) and flux MF (Figure S22) are a multi-way tie, as many models achieve a perfect MF of 1.0. In this case, the tie is broken arbitrarily and the star marks one of the many models with perfect MF. The selected model has $N_b^{\text{fully}} = 3$, $N_b^{\text{upsampling}} = 2$, $N_b^{\text{skip}} = 1$, spectral complexity of 128, and uses the reparameterization method instead of flipout. The first four of these hyperparameter values are represented in Figure S10.

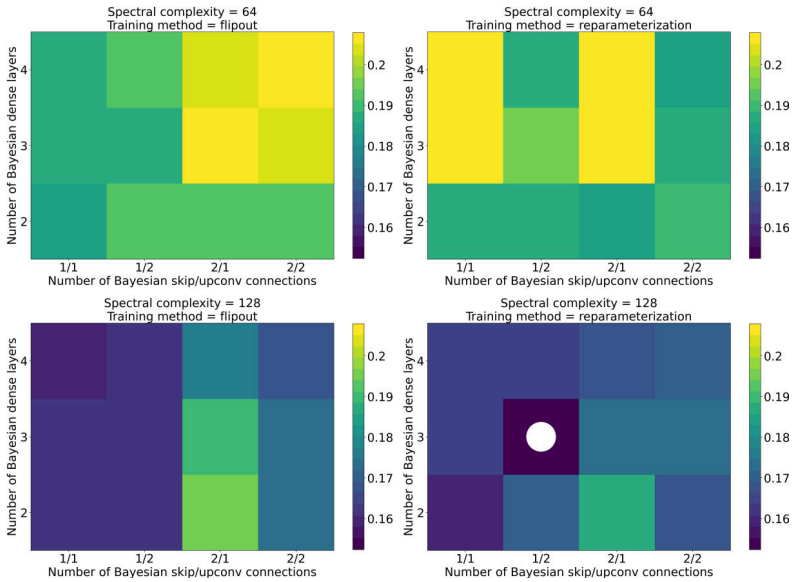


Figure S11: MAE for heating rate, computed on validation data for each set of hyperparameters. "Dense" here is a synonym for "fully connected". Each panel shows one spectral complexity and one Bayesian training method (reparameterization or flipout); within each panel the other three hyperparameters vary. The white circle marks the selected model, and the white star (hidden behind the white circle) marks the model with the lowest value for this error metric.

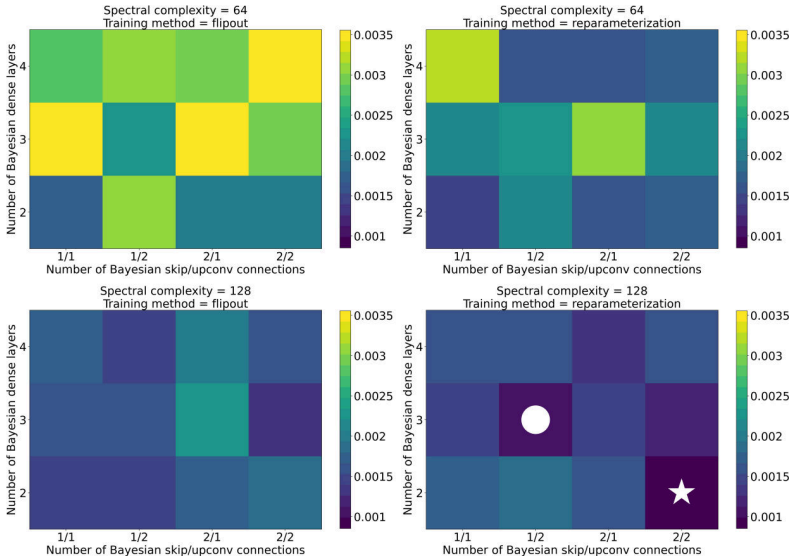


Figure S12: REL for heating rate, computed on validation data for each set of hyperparameters. Formatting is explained in the caption of Figure S11.

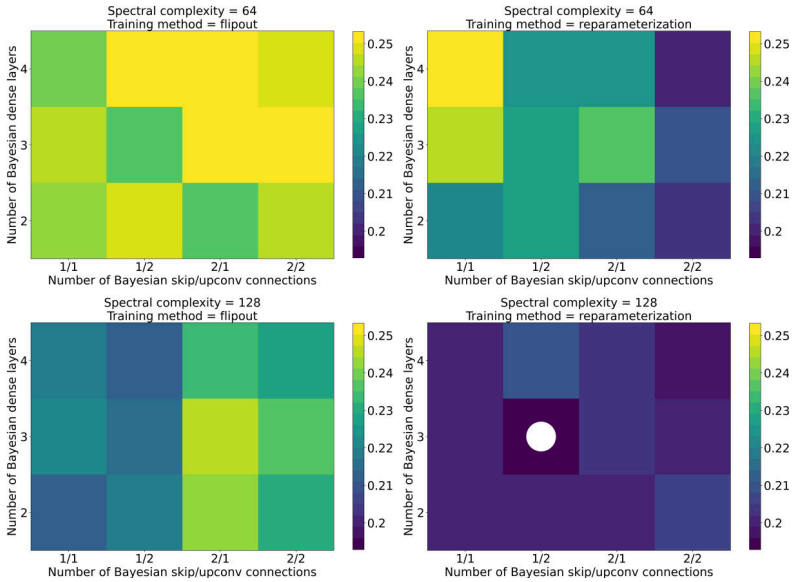


Figure S13: SSREL for heating rate, computed on validation data for each set of hyperparameters. Formatting is explained in the caption of Figure S11.

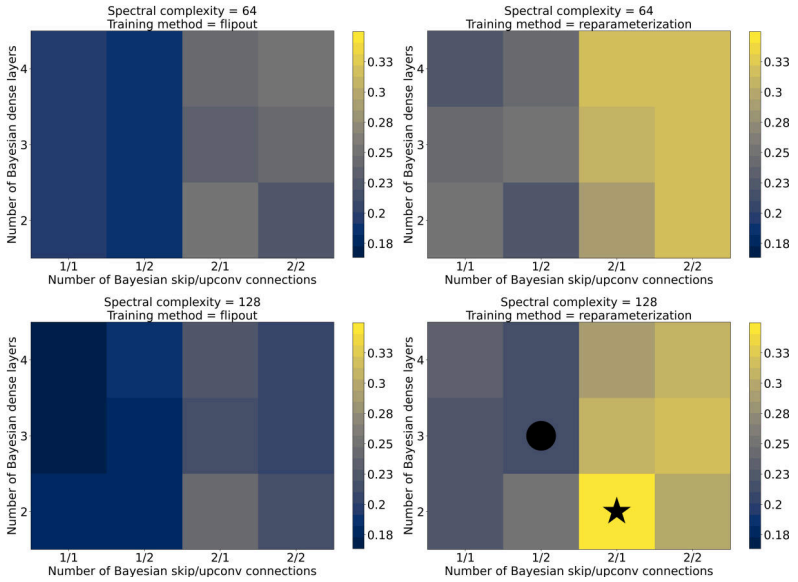


Figure S14: SSRAT for heating rate, computed on validation data for each set of hyperparameters. Formatting is explained in the caption of Figure S11.

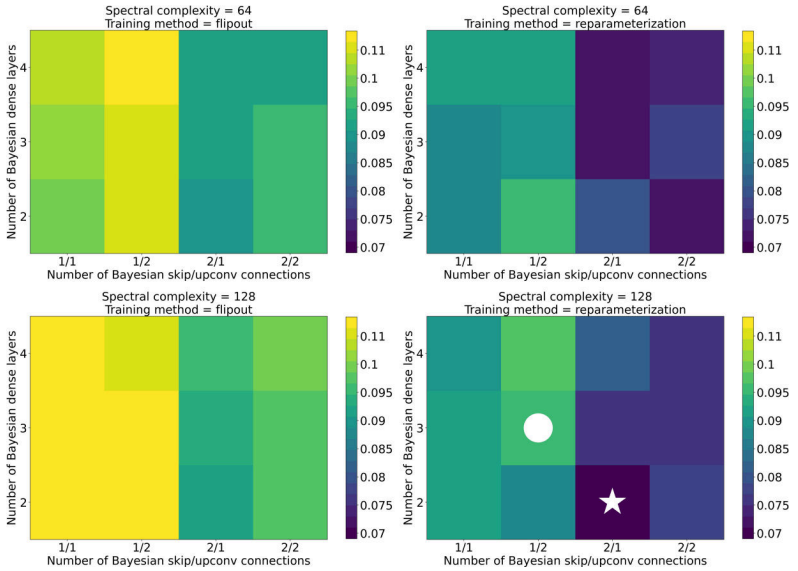


Figure S15: PITD for heating rate, computed on validation data for each set of hyperparameters. Formatting is explained in the caption of Figure S11.

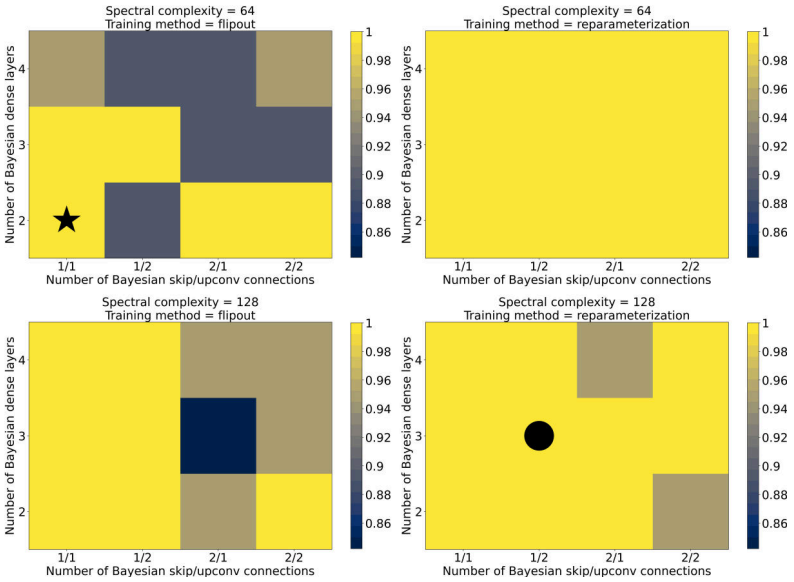


Figure S16: MF for heating rate, computed on validation data for each set of hyperparameters. Formatting is explained in the caption of Figure S11.

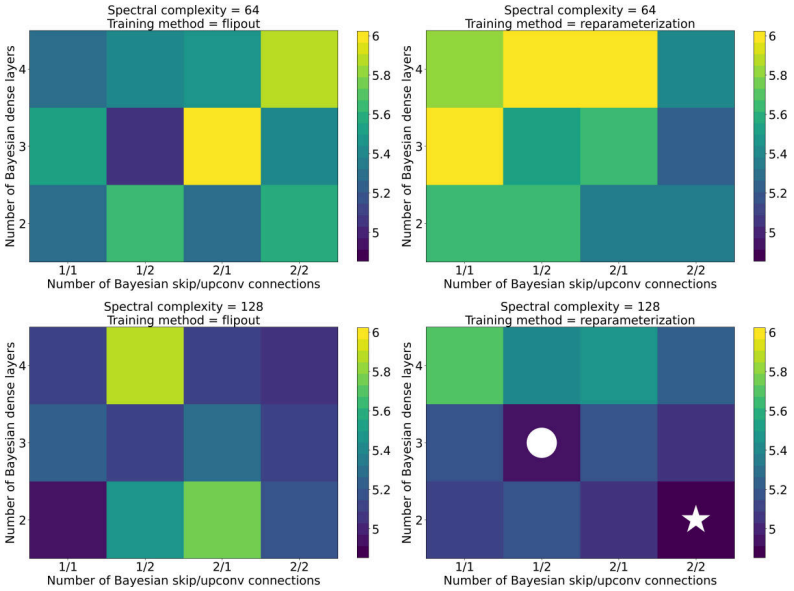


Figure S17: MAE for flux variables, computed on validation data for each set of hyperparameters. Formatting is explained in the caption of Figure S11.

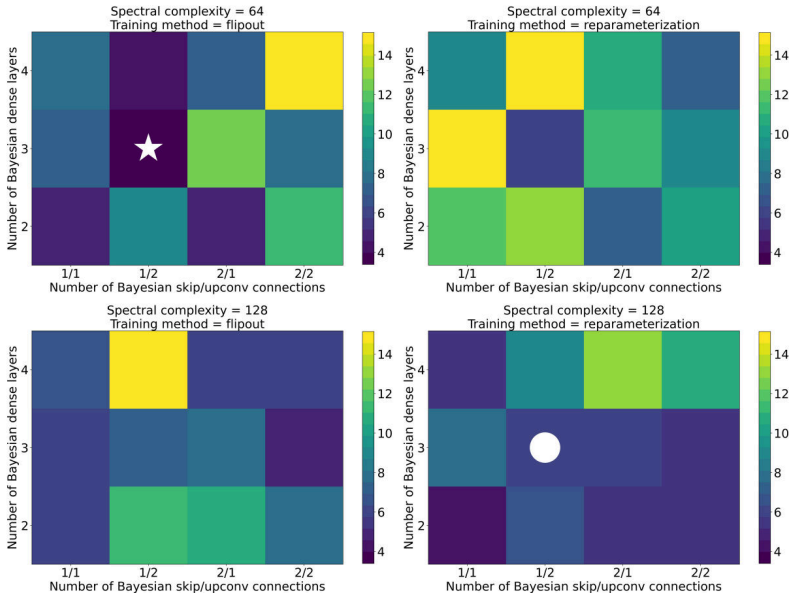


Figure S18: REL for flux variables, computed on validation data for each set of hyperparameters. Formatting is explained in the caption of Figure S11.

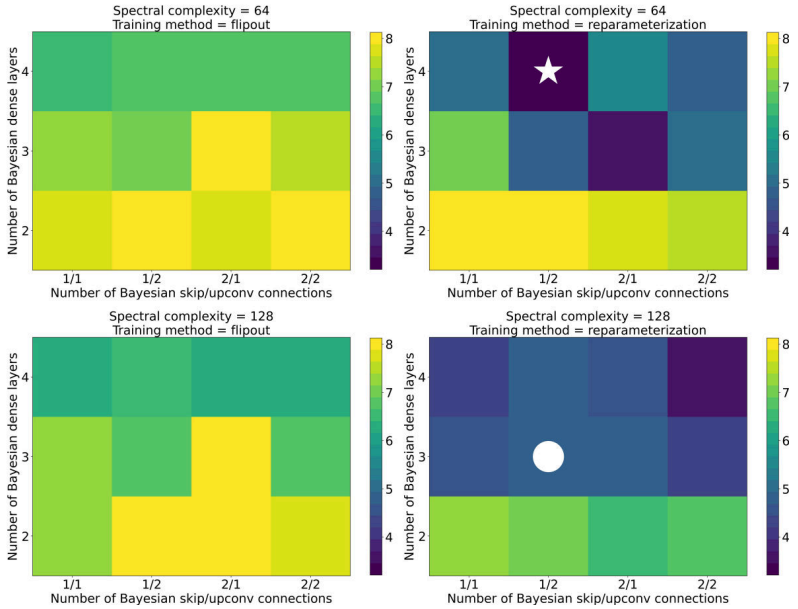


Figure S19: SSREL for flux variables, computed on validation data for each set of hyperparameters. Formatting is explained in the caption of Figure S11.

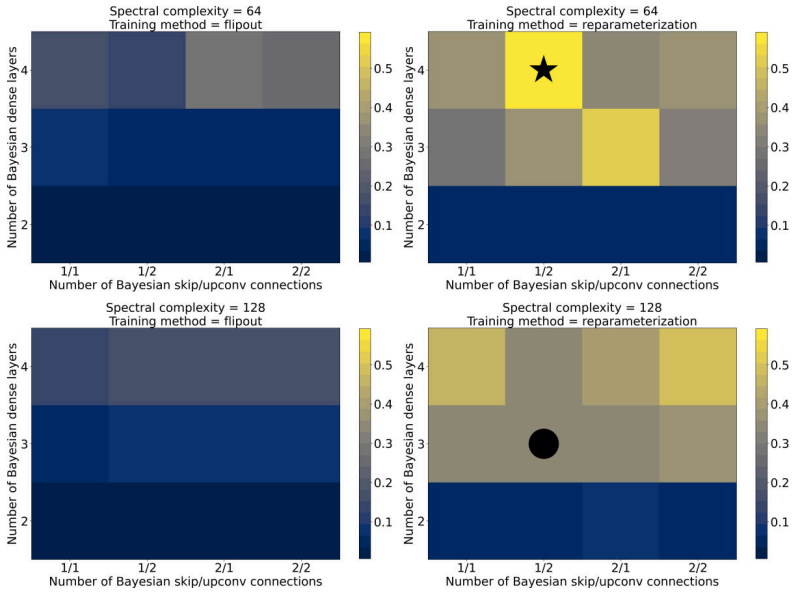


Figure S20: SSRAT for flux variables, computed on validation data for each set of hyperparameters. Formatting is explained in the caption of Figure S11.

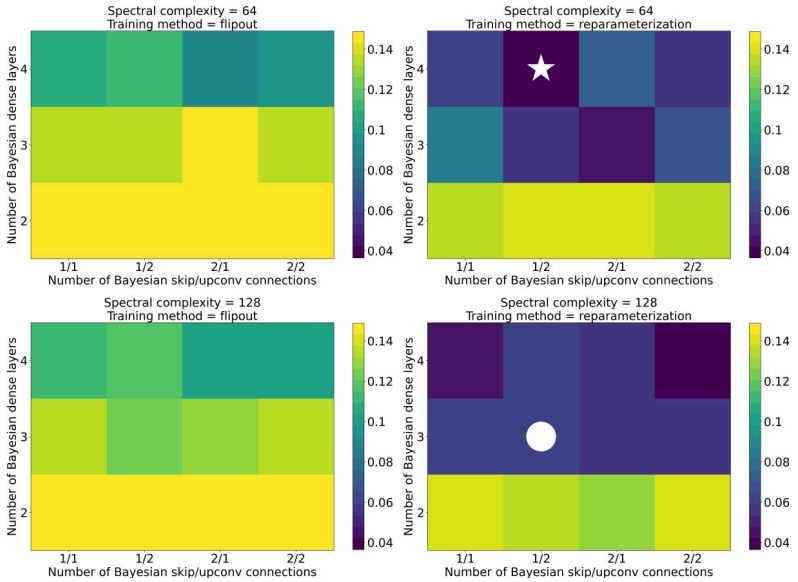


Figure S21: PITD for flux variables, computed on validation data for each set of hyperparameters. Formatting is explained in the caption of Figure S11.

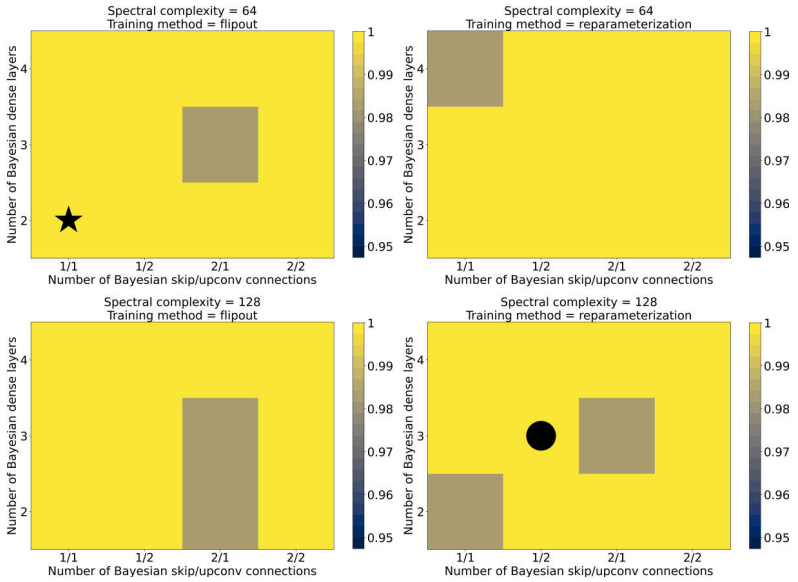


Figure S22: MF for flux variables, computed on validation data for each set of hyperparameters. Formatting is explained in the caption of Figure S11.

4 Further results of Experiment 1 (clean training data)

Figure S23 shows error metrics, based on the *heavily perturbed* testing data, for all five UQ methods. This is analogous to Figure 5 in the main text, which shows results on the *moderately perturbed* validation data. The main purpose of Figure S23 is to show that the overall ranking of UQ methods is similar between the validation and testing data. Most importantly, BNN/CRPS appears to be the best method for both datasets.

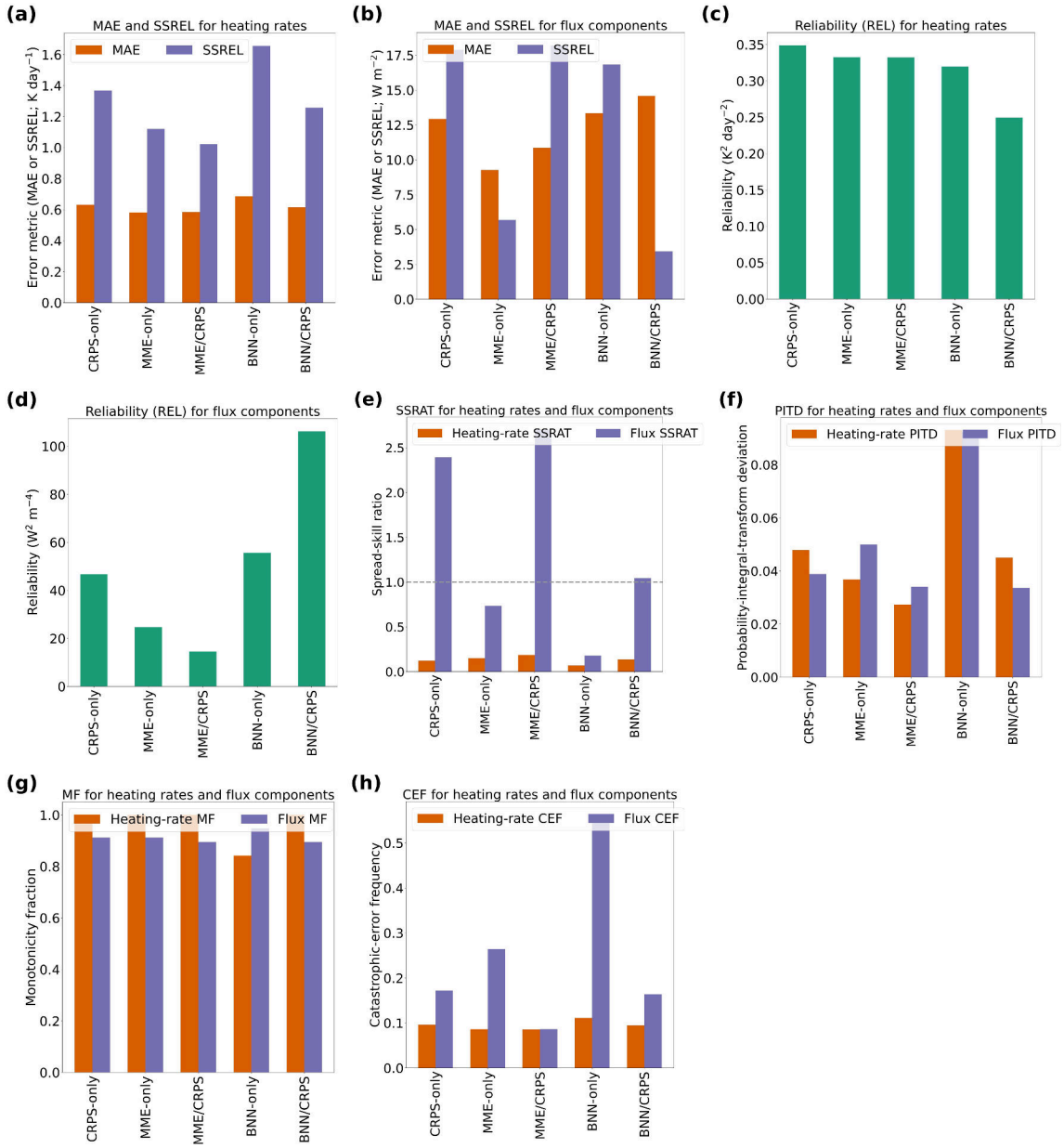


Figure S23: Comparison of UQ methods on testing data, for models trained with clean (unperturbed) data. In panel g, higher is better; in panel e, closer to 1.0 is better; in all other panels, lower is better. "CEF" in panel h is catastrophic-error frequency.

Figure S24 shows detailed results for the BNN/CRPS model on the subset of the testing data with perturbed near-surface temperature. This is analogous to Figure 7 in the main text, which shows results on the entire testing set. Figures S25-S28 are analogous to S24 but for different perturbation types: near-surface moisture, liquid cloud, ice cloud, and ozone. Some broad conclusions from these figures are highlighted in the main text.

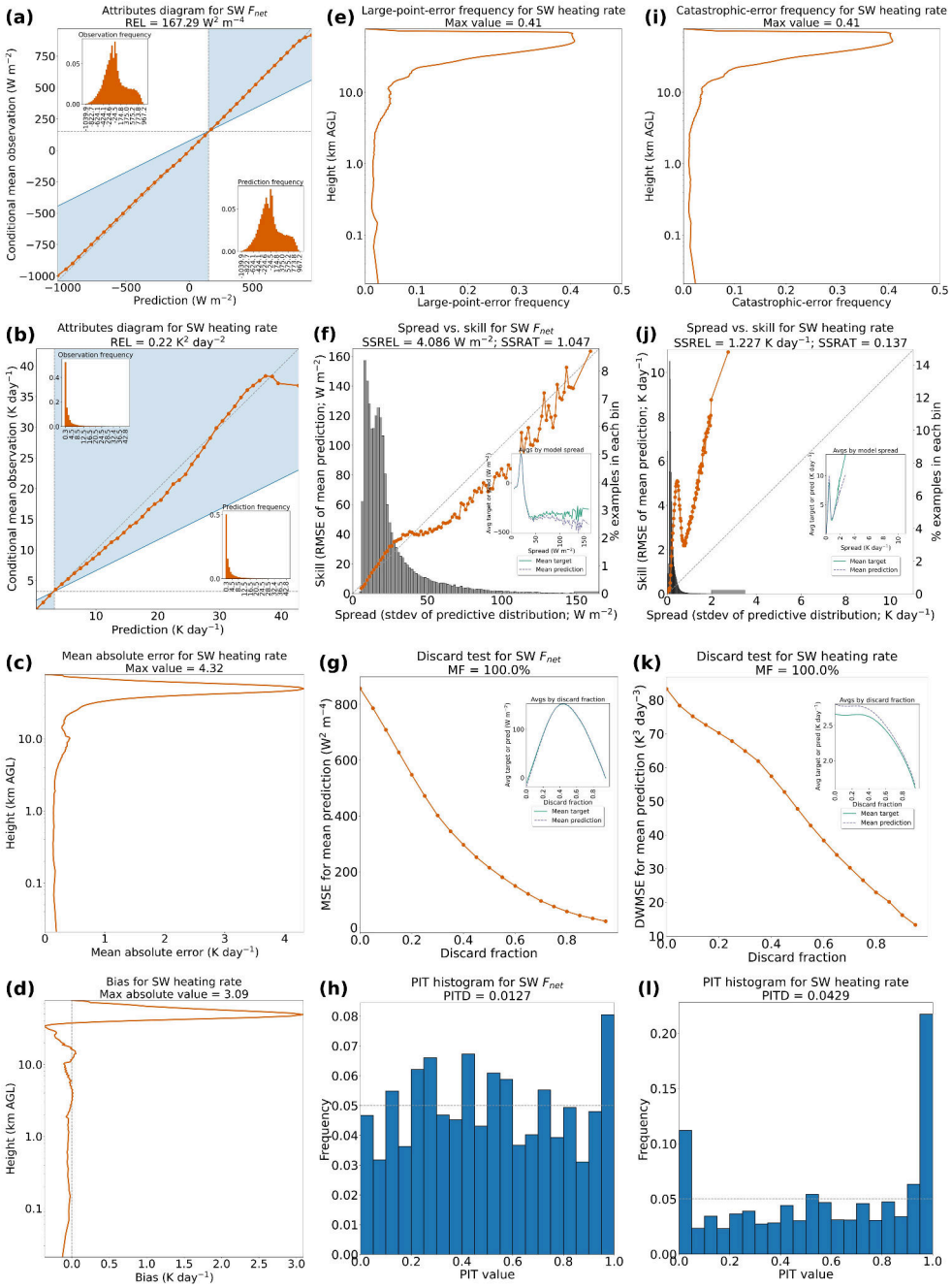


Figure S24: Detailed results of the BNN/CRPS method, for a model trained with clean data, on the subset of the testing data with perturbed near-surface temperature. Formatting is explained in the caption of Figure 6 in the main text.

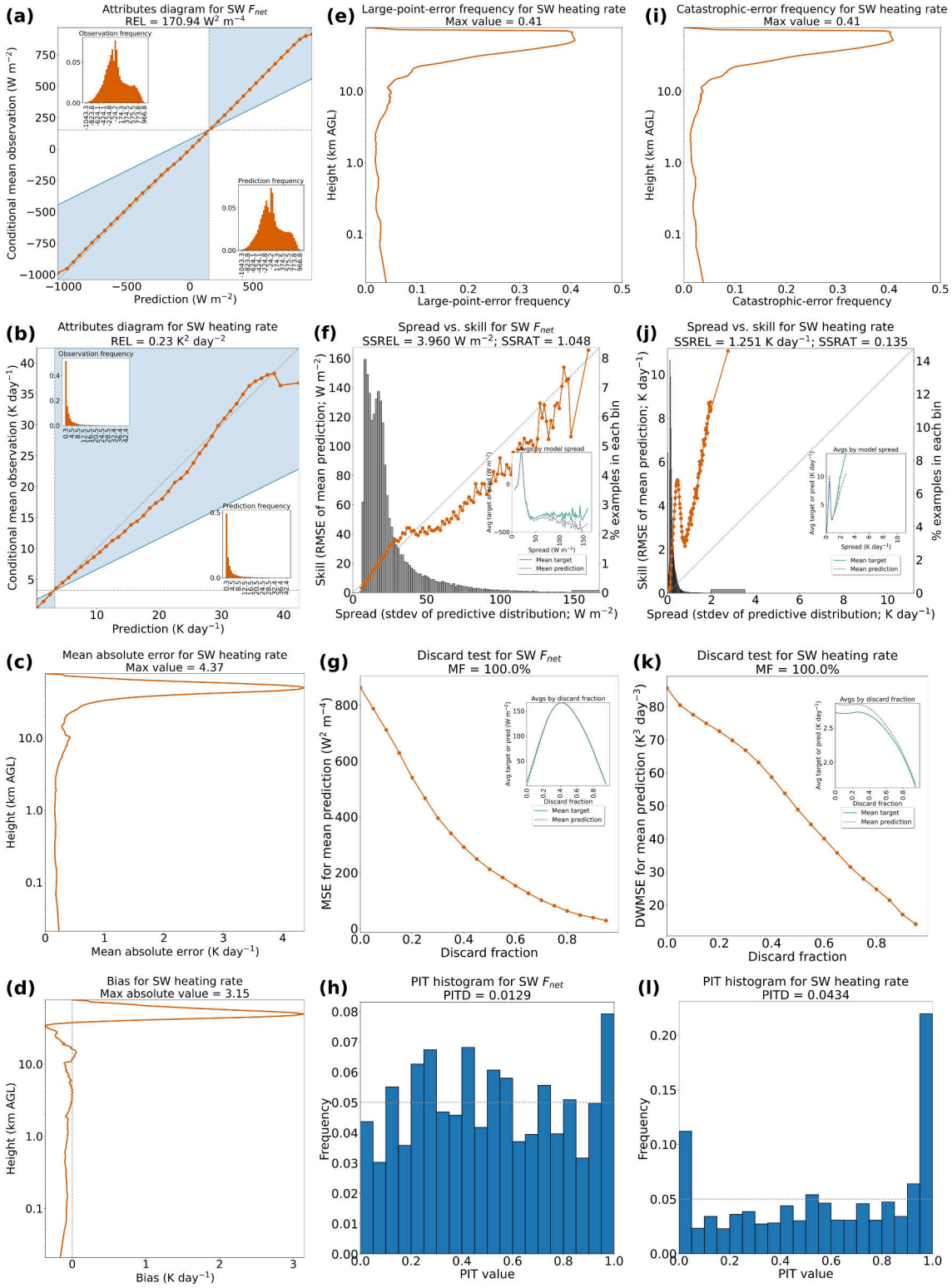


Figure S25: Same as Figure S24 but for the subset of testing data with perturbed near-surface moisture.

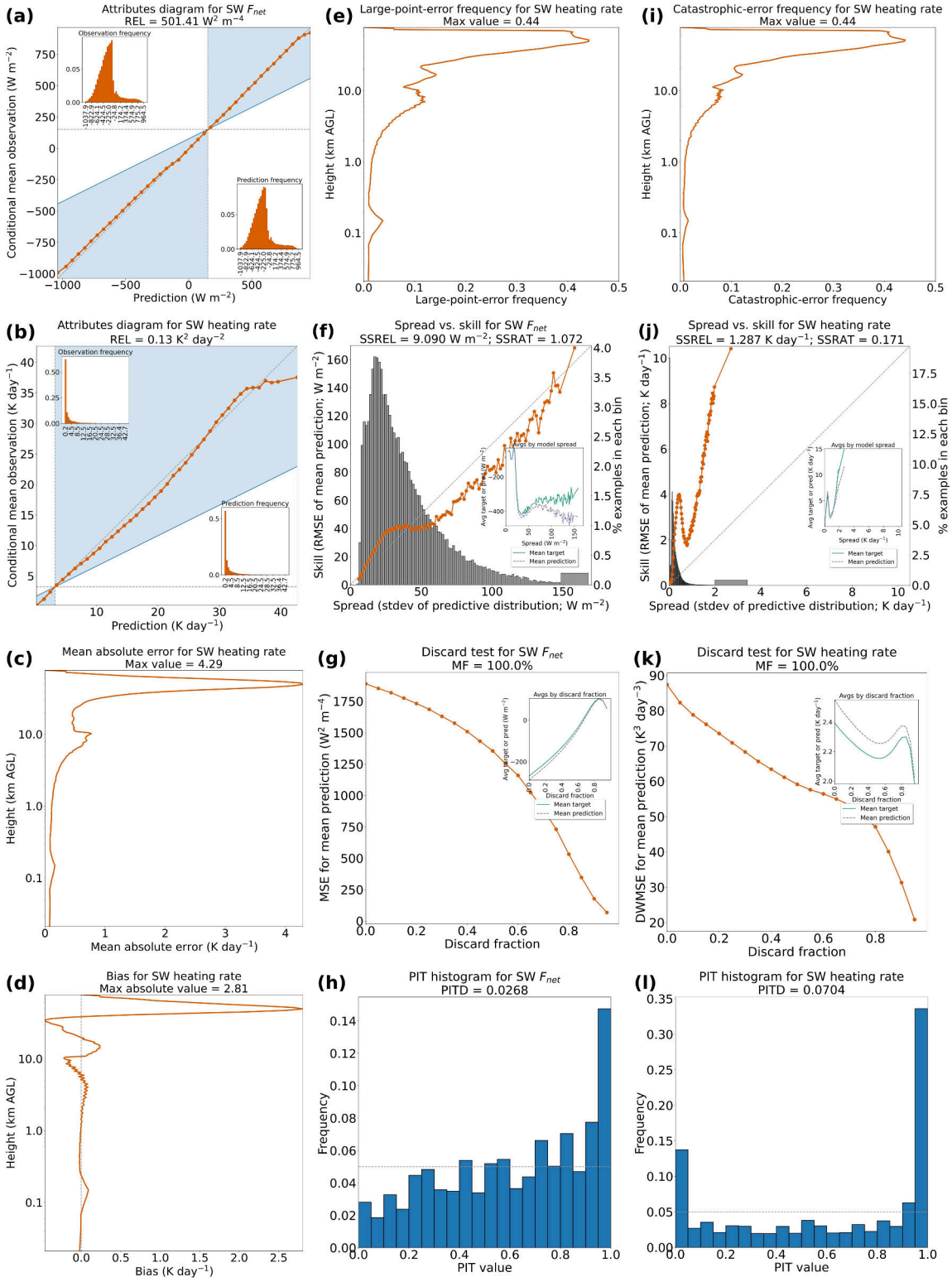


Figure S26: Same as Figure S24 but for the subset of testing data with perturbed liquid cloud.

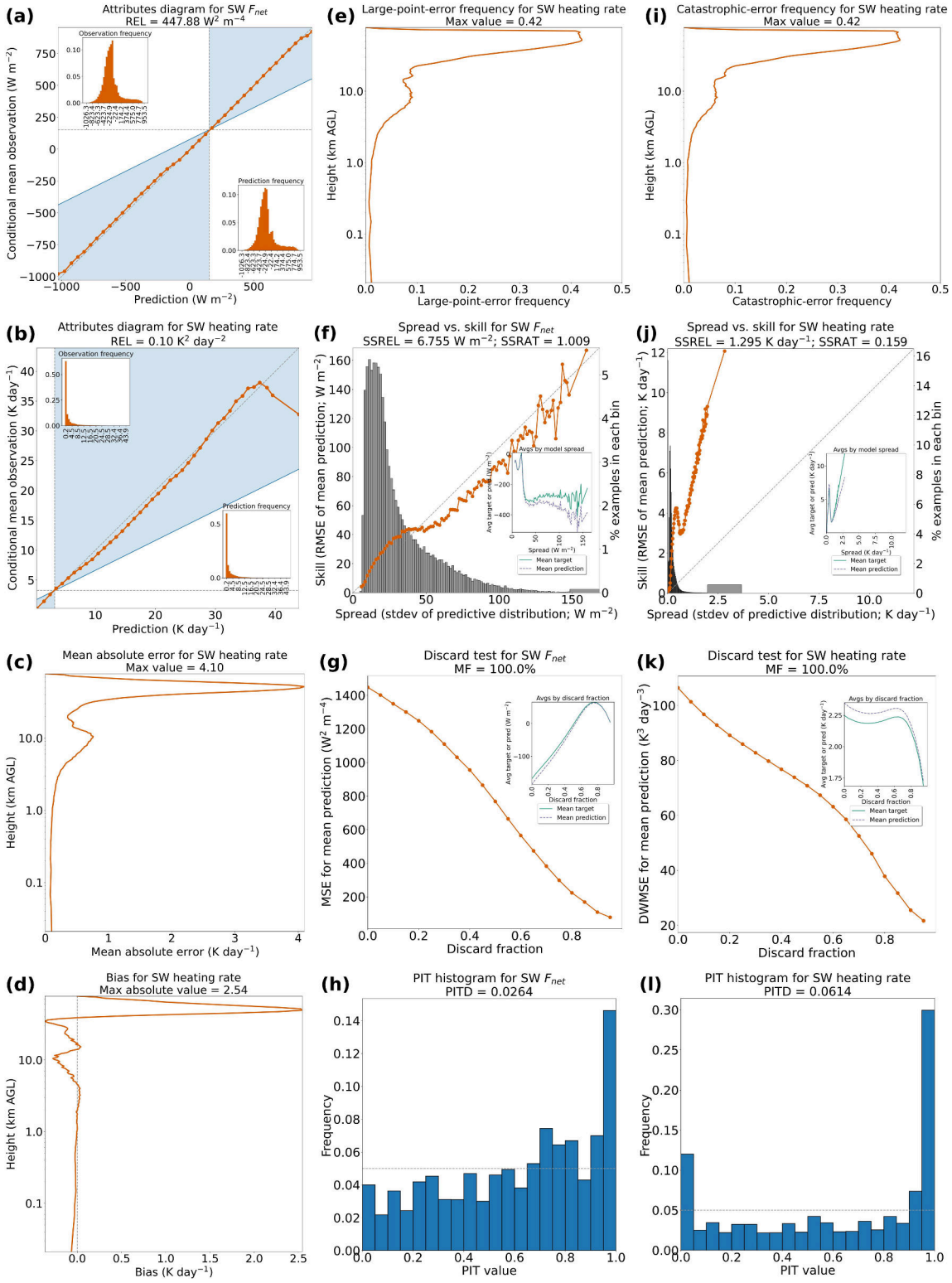


Figure S27: Same as Figure S24 but for the subset of testing data with perturbed ice cloud.

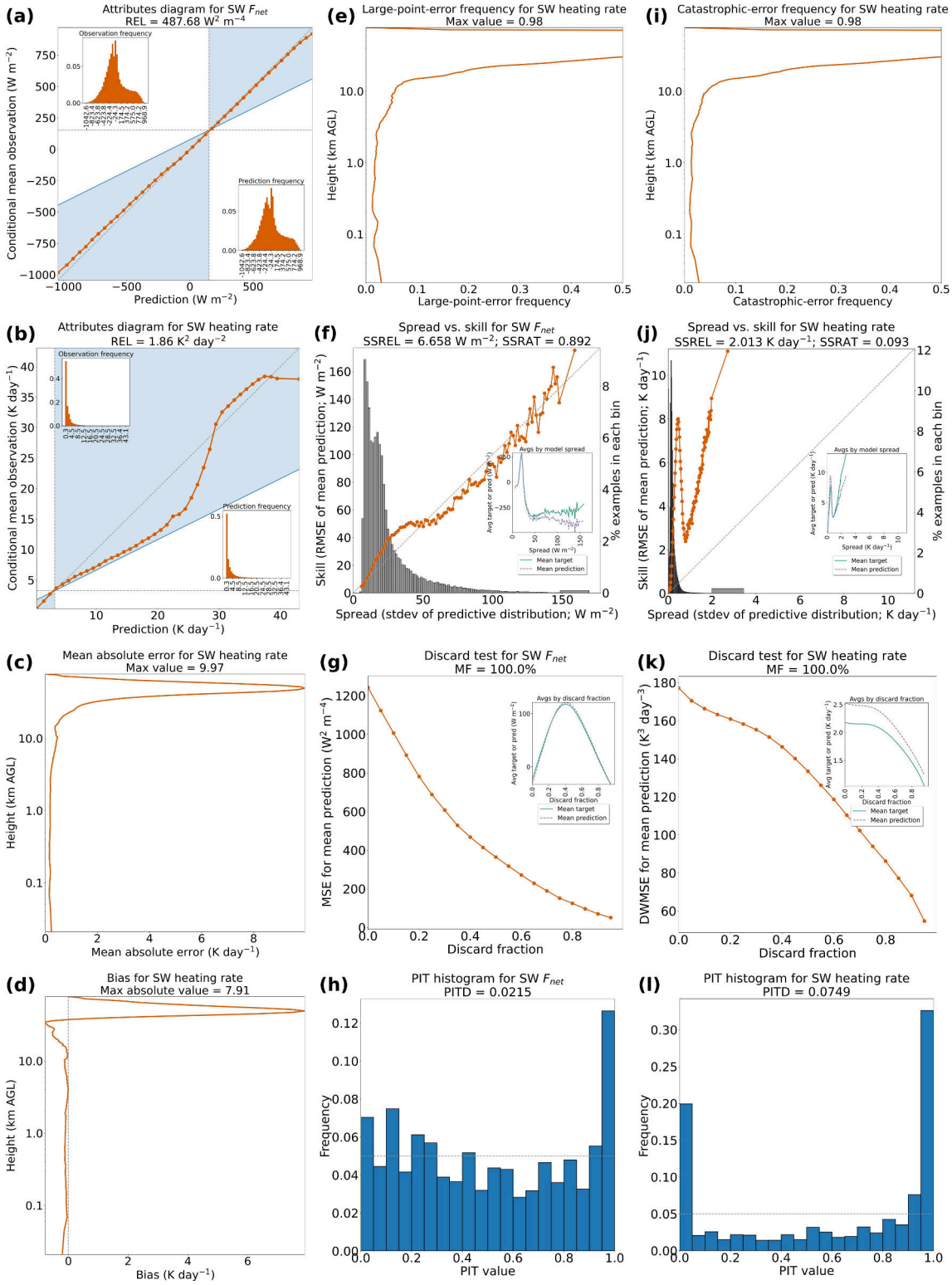


Figure S28: Same as Figure S24 but for the subset of testing data with perturbed ozone.

5 Further results of Experiment 2 (lightly perturbed training data)

Figure S29 shows error metrics, based on the *heavily perturbed* testing data, for all five UQ methods. This is analogous to Figure 11 in the main text, which shows results on the *moderately perturbed* validation data. The main purpose of this new figure is to show that the BNN/CRPS method appears to be best for both datasets.

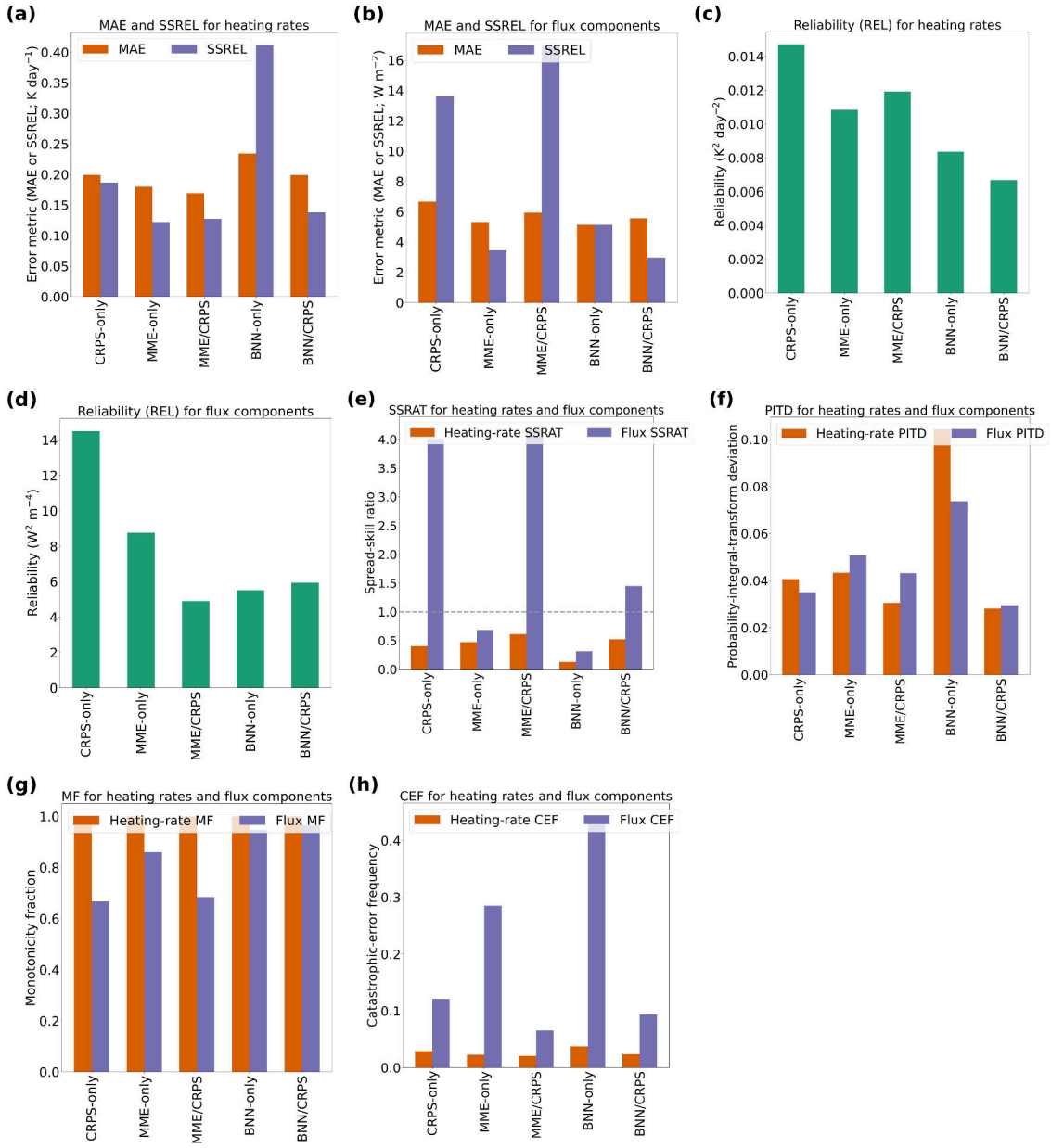


Figure S29: Comparison of UQ methods on testing data, for models trained with lightly perturbed data. In panel g, higher is better; in panel e, closer to 1.0 is better; in all other panels, lower is better. "CEF" in panel h is catastrophic-error frequency.

Figure S30 shows detailed results for the BNN/CRPS model on the validation data. This is analogous to Figure 12 in the main text, which shows results on the testing set.

The main purpose of this new figure is to show that concerning properties of the testing results are much less concerning in the validation results. Specifically, the positive bias for large HR (when ensemble mean $\gtrsim$ 38 K day^{-1}) decreases from $\sim$5 K day^{-1} to $\sim$1 K day^{-1}; struggles with perturbed ozone improve (catastrophic-error frequency in the upper stratosphere decreases from 14% to 5%); and overall underestimation of HR uncertainty improves, with SSRAT increasing from 0.520 to 0.884.

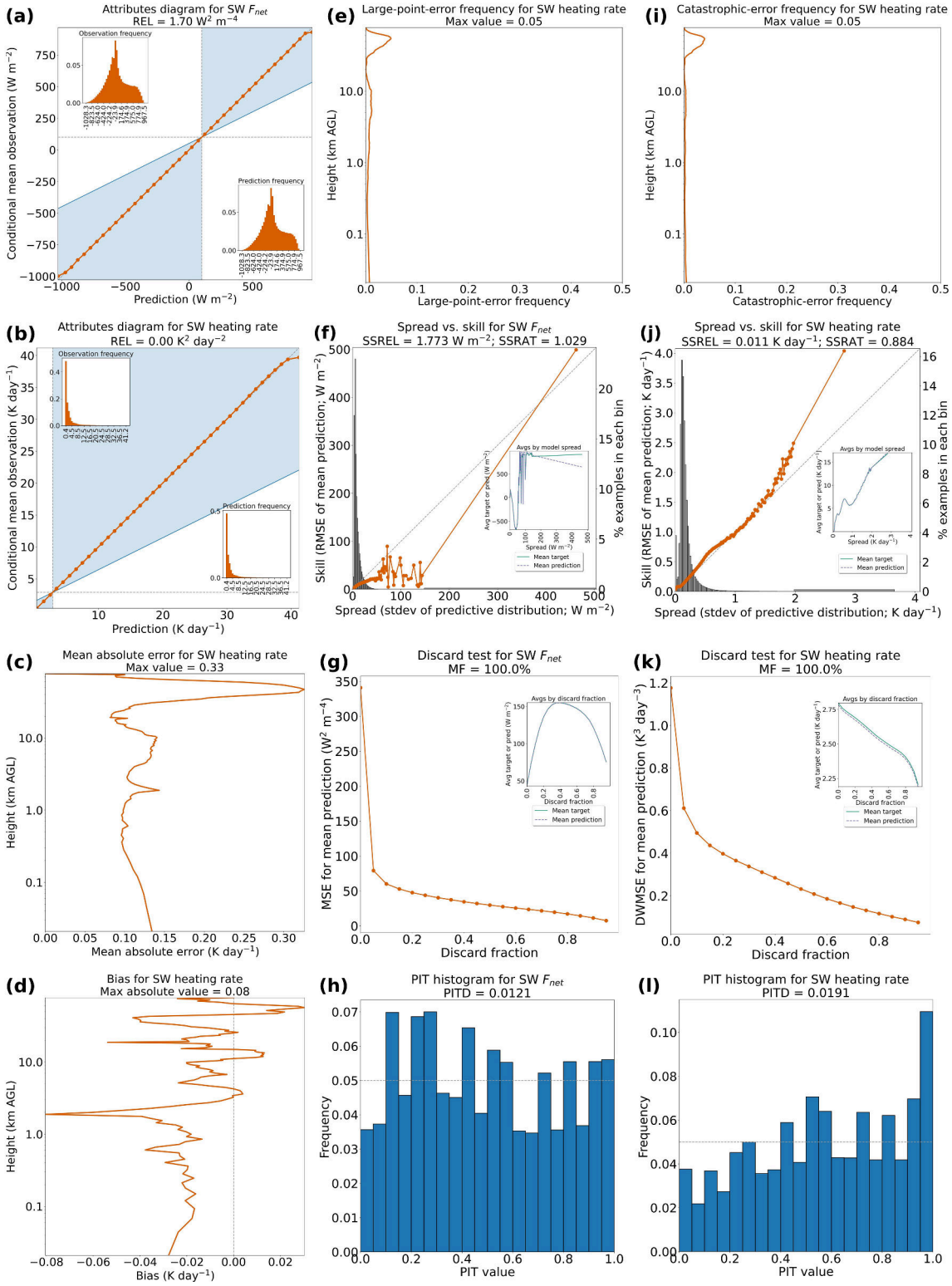


Figure S30: Detailed results of the BNN/CRPS method, for a model trained with lightly perturbed data, on the validation data. Formatting is explained in the caption of Figure 6 in the main text.

Figure S31 shows detailed results for the BNN/CRPS model on the subset of the testing data with perturbed near-surface temperature. This is analogous to Figure 12 in the main text, which shows results on the entire testing set. Figures S32-S35 are analogous to S31 but for different perturbation types: near-surface moisture, liquid cloud, ice cloud, and ozone. Some broad conclusions from these figures are highlighted in the main text.

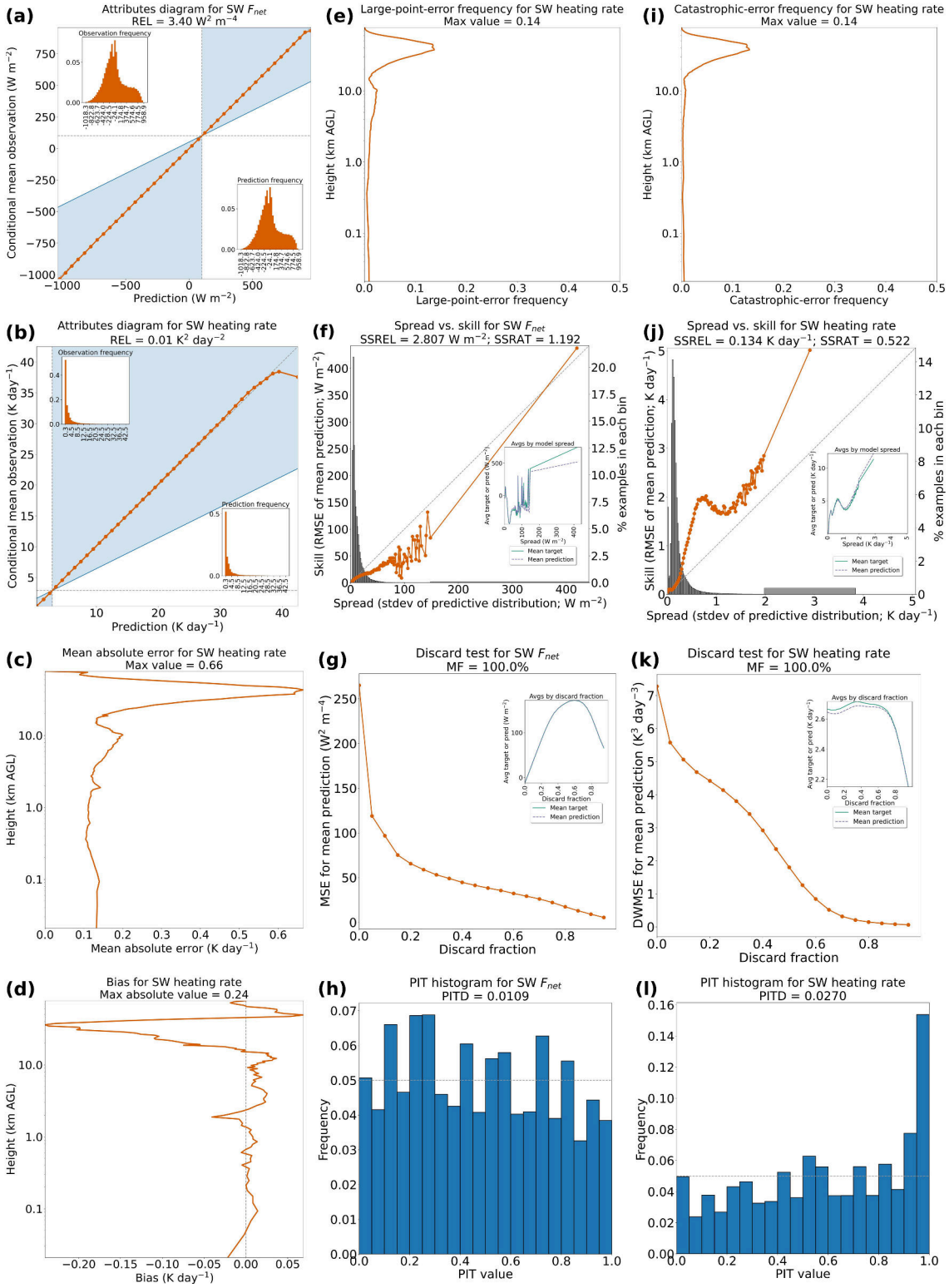


Figure S31: Detailed results of the BNN/CRPS method, for a model trained with lightly perturbed data, on the subset of the testing data with perturbed near-surface temperature. Formatting is explained in the caption of Figure 6 in the main text.

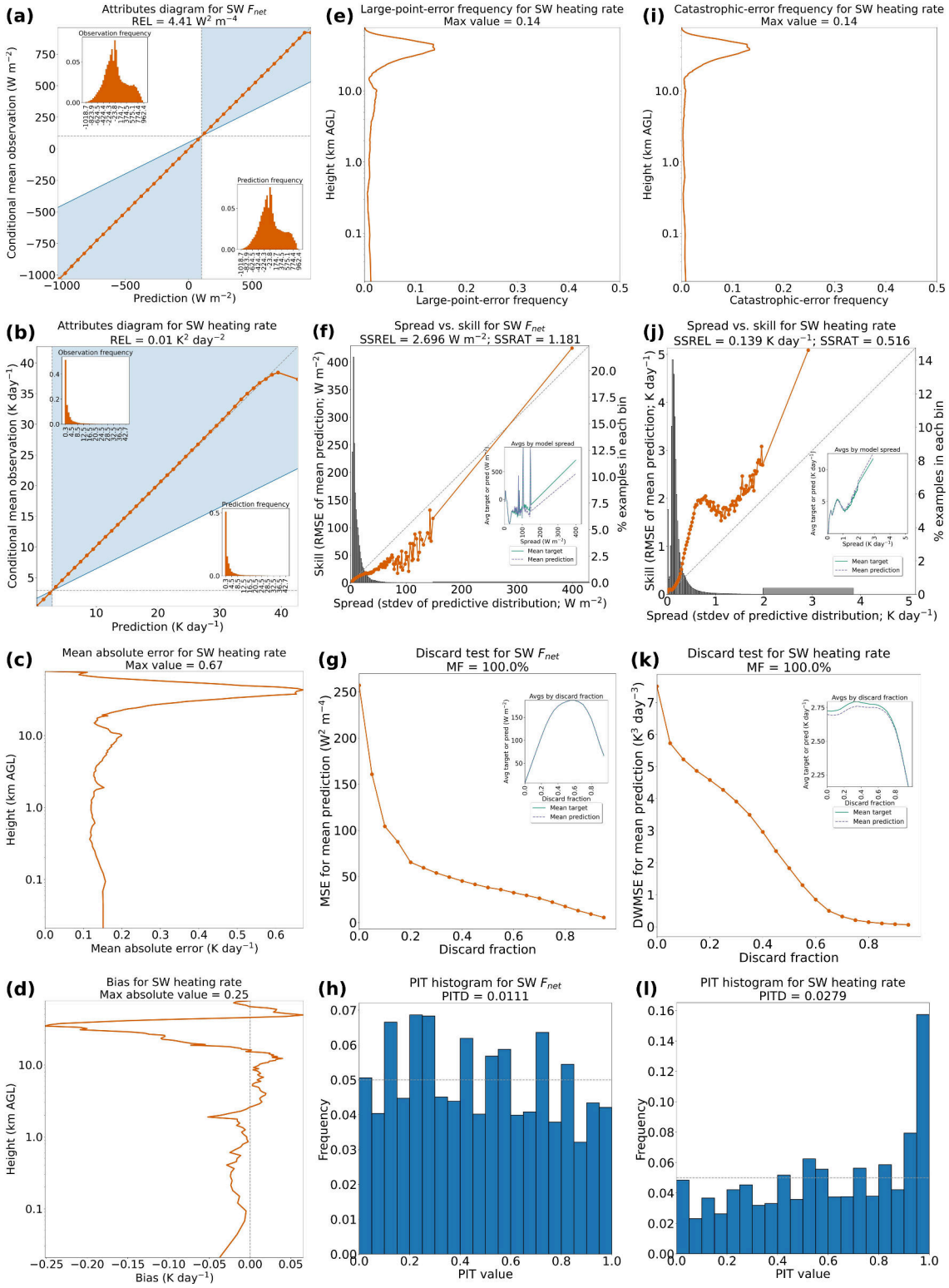


Figure S32: Same as Figure S31 but for the subset of testing data with perturbed near-surface moisture.

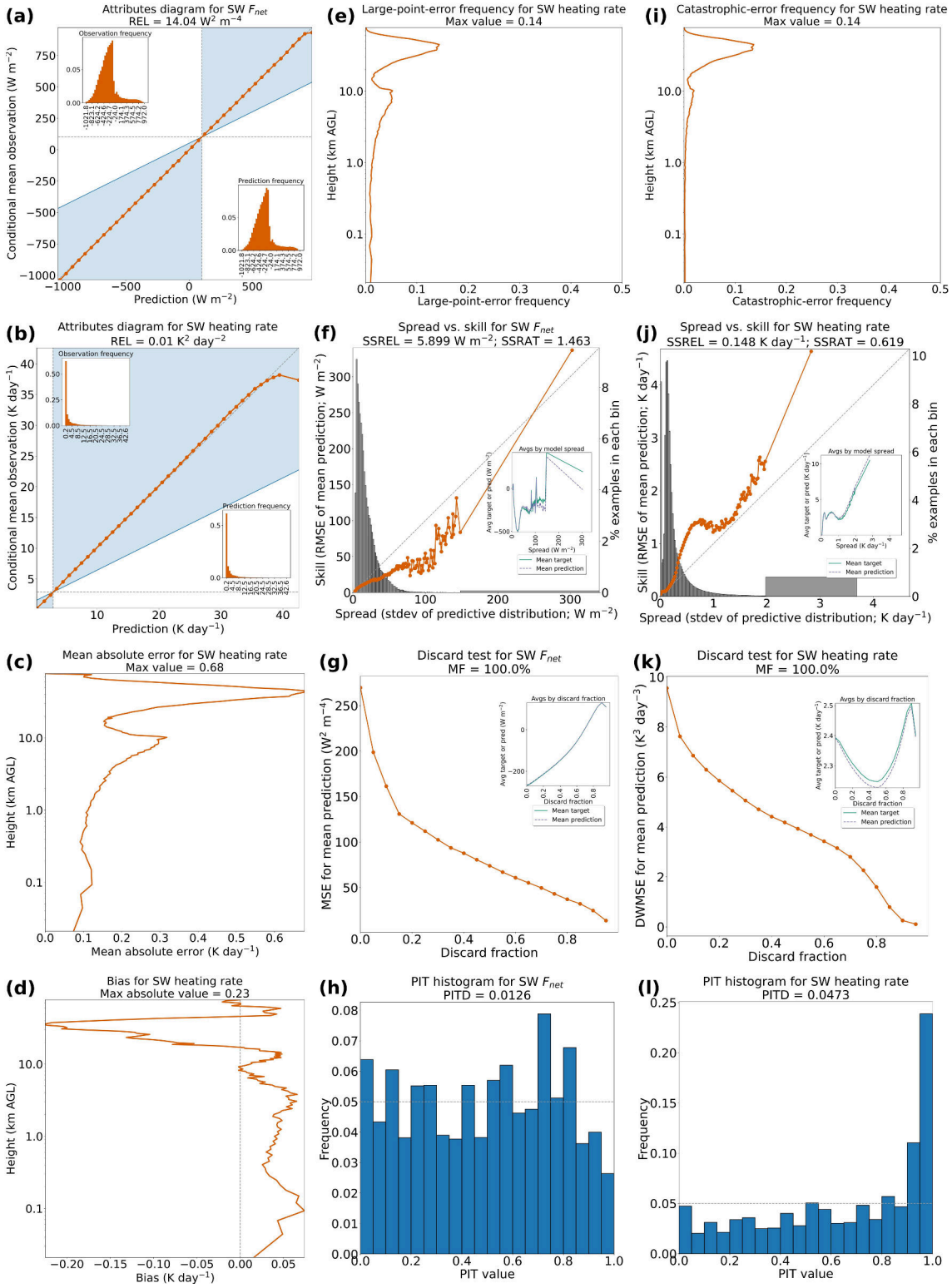


Figure S33: Same as Figure S31 but for the subset of testing data with perturbed liquid cloud.

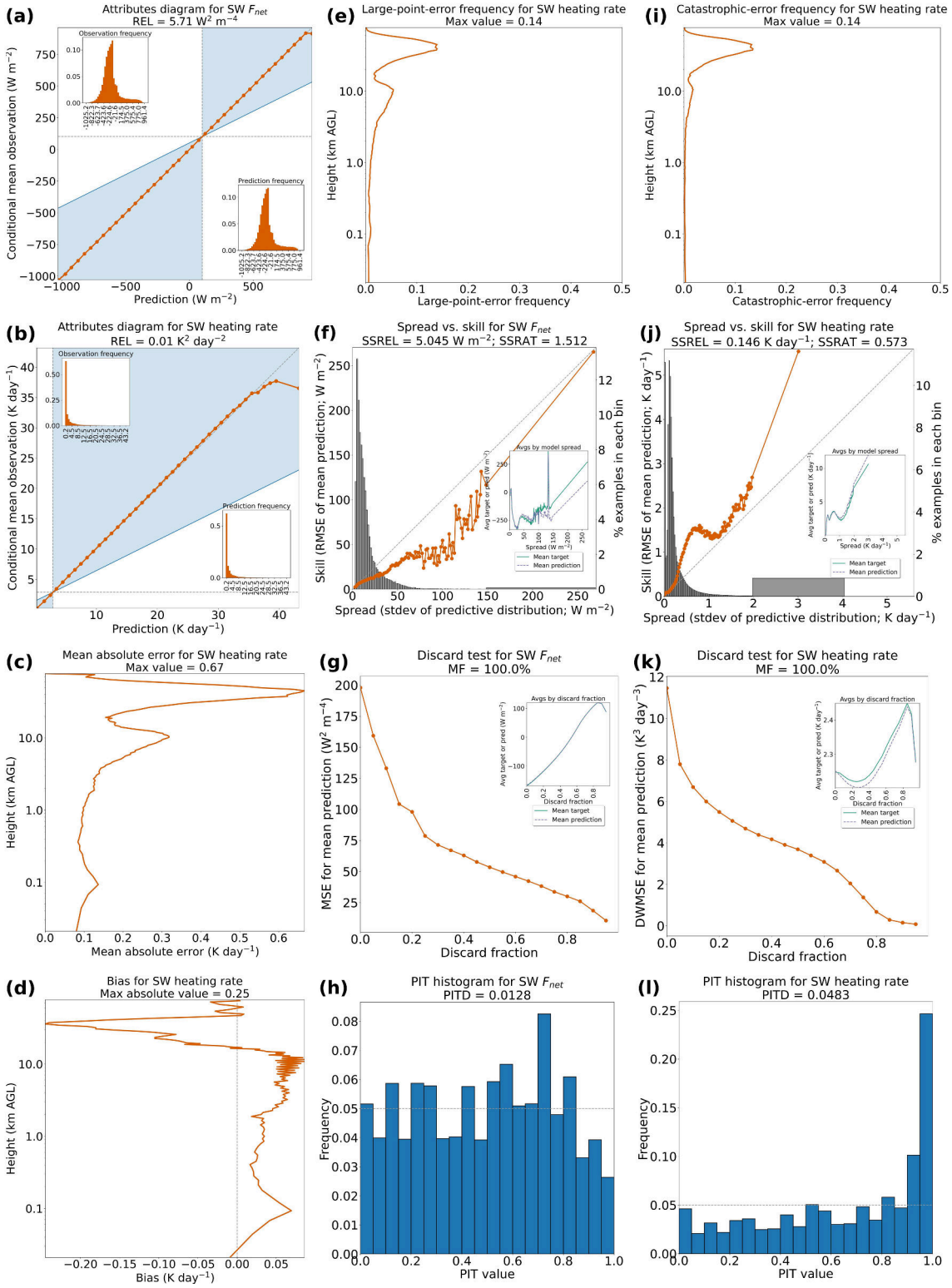


Figure S34: Same as Figure S31 but for the subset of testing data with perturbed ice cloud.

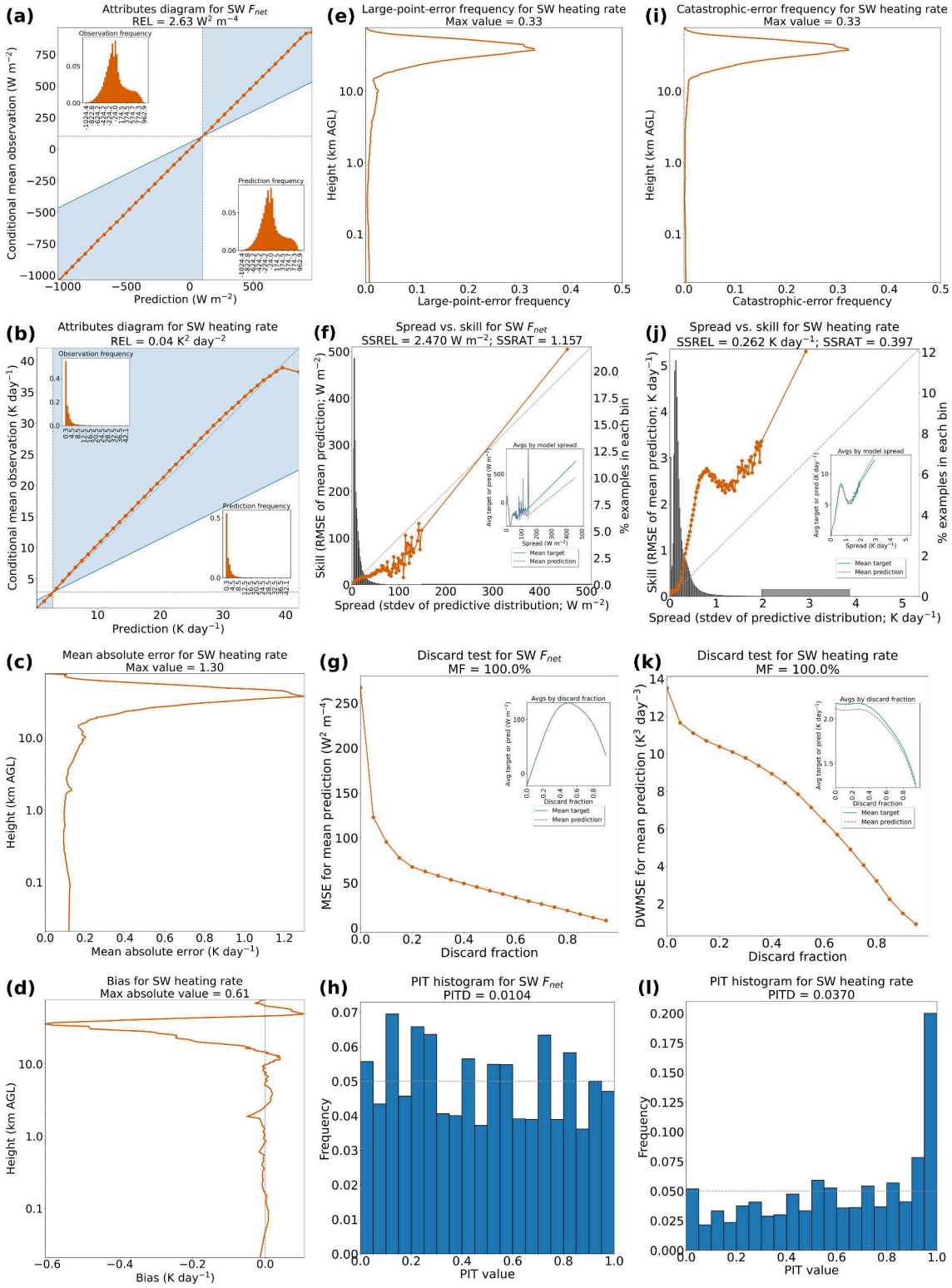


Figure S35: Same as Figure S31 but for the subset of testing data with perturbed ozone.

References

Cooney, J., Bowman, K., Homeyer, C., & Fenske, T. (2018). Ten year analysis of tropopause-overshooting convection using GridRad data. *Journal of Geophysical Research: Atmospheres*, *123*(1), 329-343. Retrieved from `https://doi.org/10.1002/2017JD027718`

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press. Retrieved from `https://www.deeplearningbook.org`

Lagerquist, R., Turner, D., Ebert-Uphoff, I., & Stewart, J. (2023). Estimating full longwave and shortwave radiative transfer with neural networks of varying complexity. *Journal of Atmospheric and Oceanic Technology*, *conditionally accepted*. Retrieved from `https://doi.org/10.22541/essoar.168319865.58439449/v1`

Slovnik, W. (1992). *International Meteorological Vocabulary* (Tech. Rep.). World Meteorological Organization.

Wallace, J., & Hobbs, P. (2006). *Atmospheric Science: An Introductory Survey* (Vol. 2). Elsevier.