

ML core meeting
Oct 13, 2021

Uncertainty Quantification Methods for Neural Networks (Part 1)

Ryan Lagerquist
Marie McGraw
Imme Ebert-Uphoff

Overview

Part 1 (today):

- 1) Motivation
- 2) Types of Uncertainty
- 3) Evaluation Criteria
- 4) Four methods:
 - The analogue ensemble (aka Delle Monache's approach)
 - The Barnes approach
 - Monte Carlo Dropout (can be interpreted as Bayesian approach) with CPRS loss function
 - Bayesian Neural Network - Bayes by Backpropagation (another Bayesian approach)

Part 2 (next time, hopefully!):

- 5) Papers that already use UQ for weather and climate (mainly 2020-2021)
(Including work by Barnes, D.J. Gagne/Foster, Will Chapman, Thomas Vandal, Orescanin)
- 6) Code examples: How-to examples for some of these methods

1) Motivation

Why do we want neural networks to provide uncertainty estimates?

1. Obvious reason:

We want to know how much we can trust a NN's answer for a specific sample.

2. Nice side effect:

A NN that knows about its own uncertainty often gives better predictions, too!

(We already saw that with Libby Barnes' abstention networks, but this holds for other methods, too.)

3. Bayesian Neural Networks - one UQ method - also provides framework to **better understand effect of using regularization and ensembles in NNs.**

1) Motivation

Simple example from

Chang, D.T.

Bayesian Neural Networks: Essentials.

arXiv preprint, v1, June 2021, <https://arxiv.org/abs/2106.13594>

Consider simple regression task with scalar output, i.e. predict scalar, y .

Traditional (deterministic) NN may yield as sample output for 10 samples:


```
Predicted: 5.8 - Actual: 6.0
Predicted: 5.7 - Actual: 5.0
Predicted: 5.9 - Actual: 6.0
Predicted: 6.3 - Actual: 6.0
Predicted: 6.3 - Actual: 8.0
Predicted: 5.8 - Actual: 5.0
Predicted: 4.9 - Actual: 6.0
Predicted: 5.1 - Actual: 5.0
Predicted: 6.4 - Actual: 6.0
Predicted: 5.8 - Actual: 5.0
```

Traditional (deterministic) NN may yield as sample output:

```
Predicted: 5.8 - Actual: 6.0
Predicted: 5.7 - Actual: 5.0
Predicted: 5.9 - Actual: 6.0
Predicted: 6.3 - Actual: 6.0
Predicted: 6.3 - Actual: 8.0
Predicted: 5.8 - Actual: 5.0
Predicted: 4.9 - Actual: 6.0
Predicted: 5.1 - Actual: 5.0
Predicted: 6.4 - Actual: 6.0
Predicted: 5.8 - Actual: 5.0
```

Probabilistic NN here yields mu and sigma → can calculate 95% confidence interval:

Prediction mean: 5.96,	stddev: 0.69,	95% CI: [7.32 - 4.6]	- Actual: 6.0
Prediction mean: 5.83,	stddev: 0.71,	95% CI: [7.24 - 4.43]	- Actual: 5.0
Prediction mean: 5.81,	stddev: 0.7,	95% CI: [7.17 - 4.44]	- Actual: 6.0
Prediction mean: 6.14,	stddev: 0.74,	95% CI: [7.59 - 4.69]	- Actual: 6.0
Prediction mean: 6.81,	stddev: 0.74,	95% CI: [8.26 - 5.35]	- Actual: 8.0
Prediction mean: 5.46,	stddev: 0.72,	95% CI: [6.86 - 4.05]	- Actual: 5.0
Prediction mean: 5.4,	stddev: 0.72,	95% CI: [6.81 - 4.0]	- Actual: 6.0
Prediction mean: 5.12,	stddev: 0.73,	95% CI: [6.56 - 3.69]	- Actual: 5.0
Prediction mean: 6.75,	stddev: 0.74,	95% CI: [8.19 - 5.3]	- Actual: 6.0
Prediction mean: 5.5,	stddev: 0.73,	95% CI: [6.93 - 4.07]	- Actual: 5.0



Observations:

- We get sigma value with each estimate. Tells us about confidence of NN prediction for that sample.
- One test for sigma: We can check whether actual value is within 95% CI. (But NN could cheat - just make sigma really large and actual value will always be in 95% CI. So not enough!)
- Also note: The actual estimates (prediction mean) have changed, too.
Estimate itself often *better* in probabilistic NN than in deterministic NN.

2) Types of Uncertainty

$$(\text{Total Uncertainty}) = (\text{Aleatory Uncertainty}) + (\text{Epistemic Uncertainty})$$

Aleatory uncertainty: the natural randomness in the underlying process.

- Classic Example:
 - Tossing a perfect coin (50-50 probability)
 - Even the best model of this system cannot predict outcome of tossing a coin, because of its stochastic properties.
- Irreducible: This uncertainty can be estimated, but not eliminated.
- Other names: statistical, stochastic or irreducible uncertainty

Epistemic uncertainty: the scientific uncertainty in the model of the process due to limited data and knowledge.

- Uncertainty based on our ignorance - we just do not know the system well enough.
- Classic example:
 - Training a machine learning model with few data samples.
 - Uncertainty can be reduced by feeding more data and/or adding more physical knowledge.
- Reducible: This uncertainty can be reduced with better models.
- Other names: systemic, model or reducible uncertainty.

2) Types of Uncertainty

How do these categories relate to categories for weather/climate applications?

Sample categories - uncertainty categories for clouds and climate

(Adopted from: Beucler et al., Machine Learning for Clouds and Climate, book chapter in “Clouds and Climate”, AGU Geophysical Monograph Series, <https://www.essoar.org/doi/abs/10.1002/essoar.10506925.1>)

1. Stochastic: due to internal climate variability, etc.
2. Observational: due to measurement and representation errors
3. Structural: due to incorrect model structure
4. Parametric: due to incorrect model parameters

Of the categories 1-4 above, which ones are aleatory vs. epistemic?

Reminder:

- Aleatory (stochastic) uncertainty: natural randomness in the underlying process.
- Epistemic (model) uncertainty: scientific uncertainty in the model of the process due to limited data and knowledge.

2) Types of Uncertainty

How do these categories relate to categories for weather/climate applications?

Sample categories - uncertainty categories for clouds and climate

(Adopted from: Beucler et al., Machine Learning for Clouds and Climate, book chapter in “Clouds and Climate”, AGU Geophysical Monograph Series, <https://www.essoar.org/doi/abs/10.1002/essoar.10506925.1>)

1. Stochastic: due to internal climate variability, etc.
2. Observational: due to measurement and representation errors (e.g., satellite retrieval error)
3. Structural: due to incorrect model structure (e.g., ML model type)
4. Parametric: due to incorrect model parameters (e.g., ML training)

Of the categories 1-4 above, which ones are aleatory vs. epistemic?

#1: Definitely aleatory (stochastic)

#3 and #4: Definitely epistemic (model)

#2: Seems to have both aleatory and epistemic components.
Some of it could be reduced by better knowledge of sensor system (e.g., satellite),
but some of it is inherent internal variability of sensor system.

2) Types of Uncertainty

Q: Which types of uncertainty *do* we care about?

A: Depends on application!

Scenario 1:

If you have high confidence in your NN, e.g., you have wonderful, large, labeled training data that span your entire domain fairly well, *and* you know your model is a great fit for the task.

→ epistemic uncertainty might be negligible

→ **Just care about aleatory uncertainty.**

→ Advantage: generally low complexity to calculate only aleatory uncertainty.

Example:

- many computer vision tasks assume this.

Scenario 2:

If you have safety-critical applications, and/or your data set is sparse.

→ **Definitely care about epistemic uncertainty, too.**

→ Ideally want to know total uncertainty (aleatory + epistemic).

Examples:

- self-driving cars,
- many CIRA application.

2) Types of Uncertainty

Concern: It seems that different UQ methods capture different types of uncertainty.

Methods may capture

1. Only aleatory (stochastic) uncertainty;
2. Only epistemic (model) uncertainty;
3. Both types of uncertainty.

We're still trying to understand completely which UQ methods capture which type of uncertainty, as different papers seem to provide somewhat contradicting statements. Stay tuned!

Text in green: indicates statements we still need to look into more deeply.
If any of you are experts in this area, we would love to get your input!

Before we dive into UQ methods

- We will focus here on four different methods to provide UQ estimates in NNs.
- Before we get into those methods...

We don't just want a method that give us "some UQ estimate".

We want a method that gives us "a good UQ estimate".

Questions (two versions of the same question):

- How will we know whether our UQ estimates are any good?
 - How should we select which method works best for our application?
- **Need Evaluation Criteria:**
 - **Tests and visualizations for quality of UQ estimate.**
 - **See next few slides.**

3. Evaluating probabilistic models

This section will discuss 6 evaluation tools:

1. Reliability curve and attributes diagram
 2. Brier score and Brier skill score
 3. Spread-skill plot
 4. Sharpness
 5. PIT histogram
 6. Continuous ranked probability score (CRPS)
- **Warning:** these evaluation tools do not separate aleatory from epistemic uncertainty.
 - Thus, when using these tools, you need to know *a priori* which type(s) of uncertainty you are evaluating!

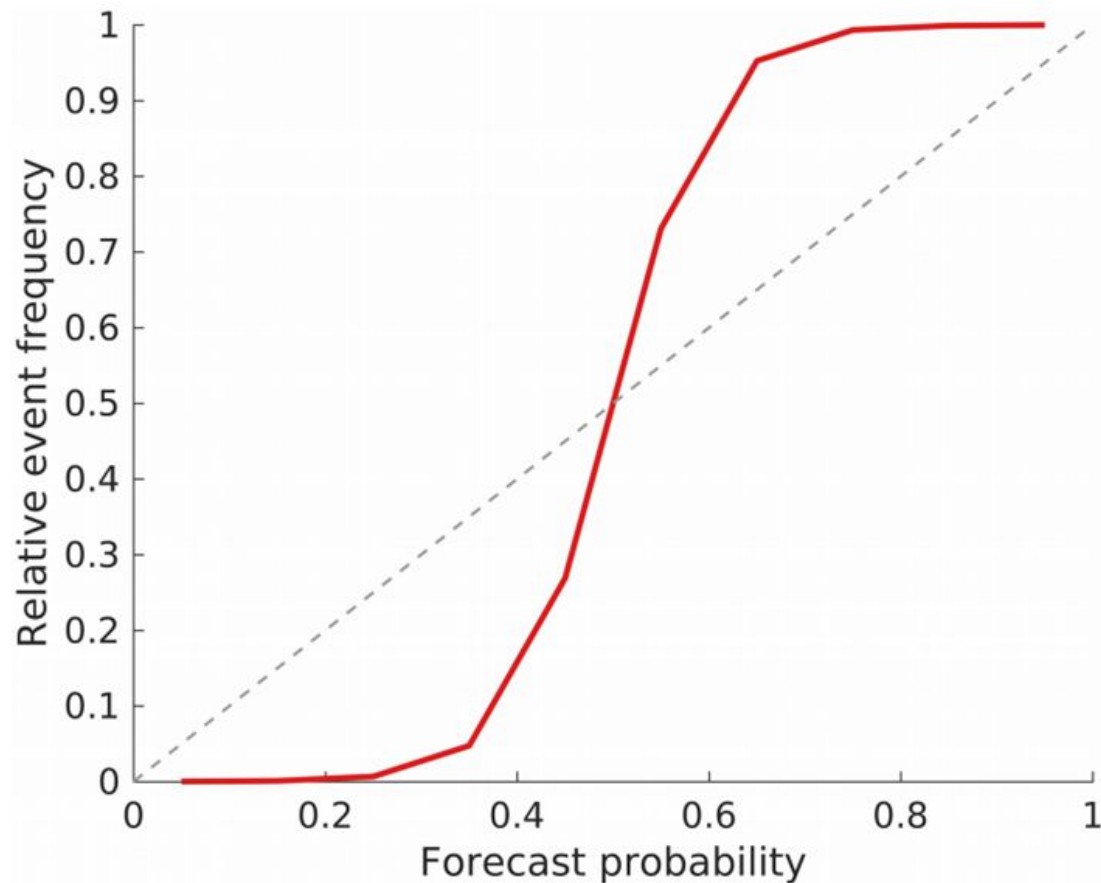
3a. Reliability curve and attributes diagram

- The reliability curve is used to evaluate predicted probabilities.
- This assumes that the task is binary classification (prediction of a yes-or-no event).
- Reliability curves are commonly used for any binary-classification model, whether the model is truly probabilistic or not.
- Most ML models for classification output confidence scores (pseudo-probabilities), which are not true probabilities.
- However, most people just call these “probabilities” and use the reliability curve to evaluate how calibrated these “probabilities” are.
- There is nothing wrong with using the reliability curve for this purpose, as long as you recognize the difference between pseudo-probabilities and true probabilities.
- Take-home point: you can use the reliability curve to evaluate true probabilities or pseudo-probabilities, but be careful with terminology.

3a. Reliability curve and attributes diagram

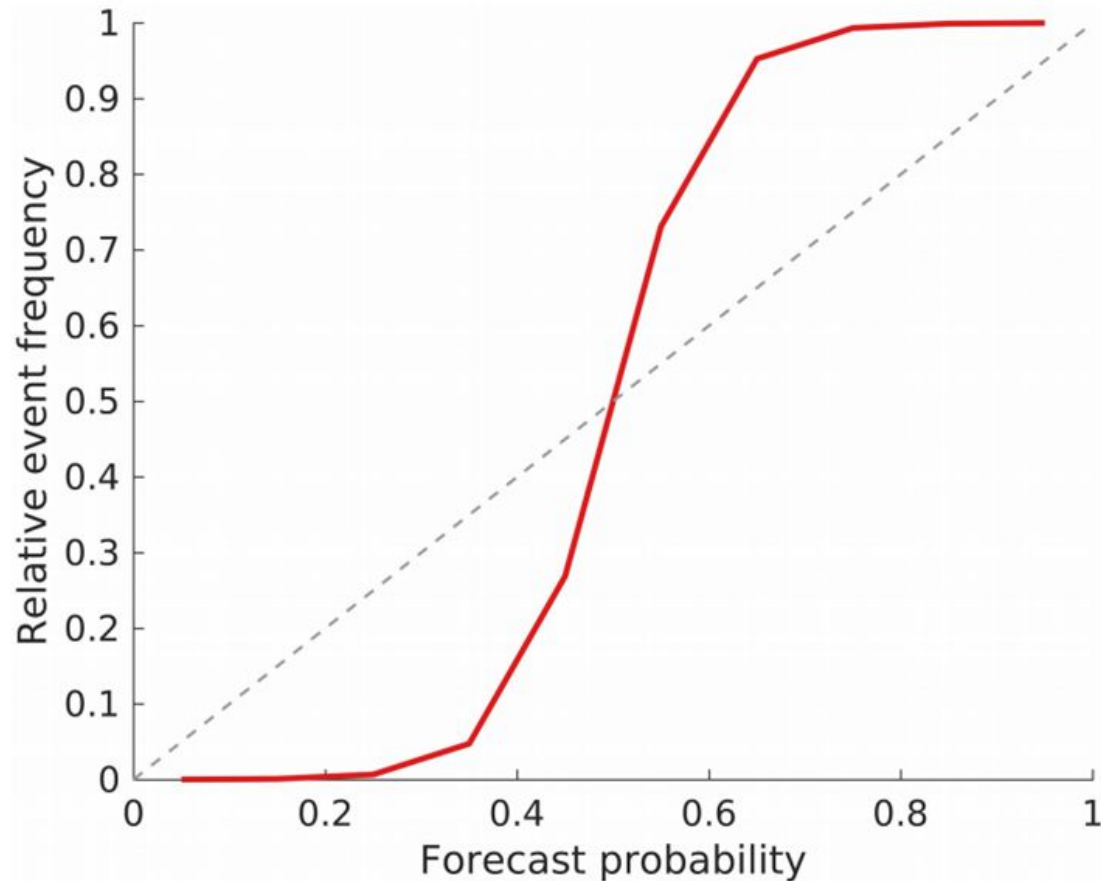
- The reliability curve plots predicted event probability vs. conditional event frequency.
- The reliability curve is binned by predicted probability, often into 10 bins:
 - 0-10%
 - 10-20%
 - ...
 - 90-100%
- Thus, each point is a mean over all examples in one bin:
 - x-coordinate: mean predicted probability in bin
 - Y-coordinate: observed event frequency in bin
- Thus, the reliability curve answers the question:

“Given predicted probability p , how likely is the event to actually occur?”



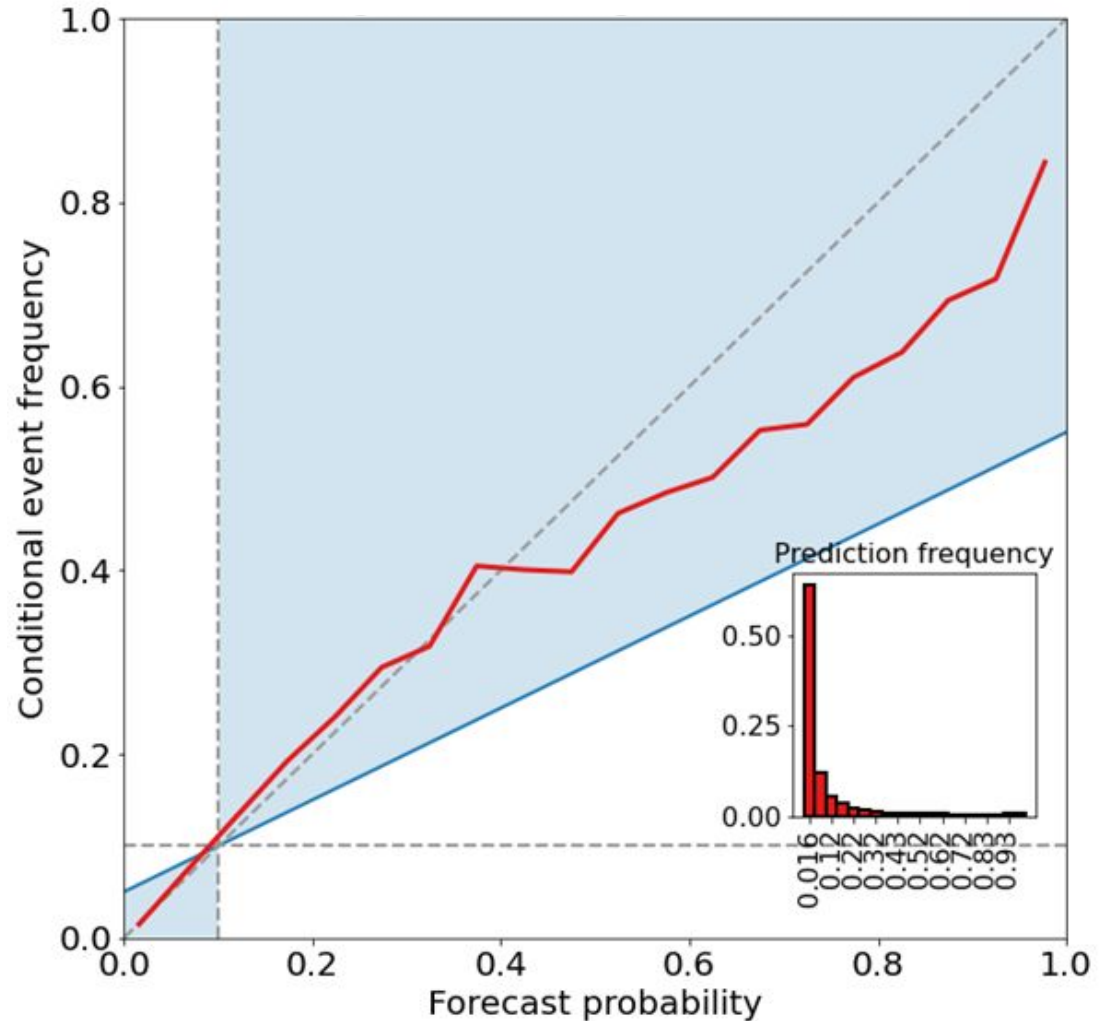
3a. Reliability curve and attributes diagram

- Ideally, conditional event frequency should always = forecast probability.
- In other words, in cases where the model says probability = p , the true event frequency (f) should be p .
- Dashed grey line: perfect reliability, where $f = p$ for all bins.
- Points below grey line: predicted probability is too high, or model is “overconfident”.
- Points above grey line: predicted probability is too low, or model is “underconfident”.



3a. Reliability curve and attributes diagram

- The attributes diagram ([Hsu and Murphy 1986](#)) is a reliability curve with extra reference lines:
 - Diagonal grey line = perfect reliability, as before
 - Vertical grey line = climatology (event frequency over all data)
 - Horizontal grey line = no resolution
 - Blue shading = positive-skill area
- A model with no resolution follows the no-resolution line.
- A climatological model (one that always predicts p = event frequency over all data) has a reliability curve with one point, at the intersection of the climo and no-resolution lines.
- “Positive skill” means Brier skill score > 0 .



3b. Brier score and Brier skill score

- The Brier score (BS) is simply the mean squared error for binary classification:

$$\text{BS} = \frac{1}{N} \sum_{i=1}^N (y_i - p_i)^2$$

- N = number of examples
 - y_i = true label (0 or 1) for i^{th} example
 - p_i = predicted event probability for i^{th} example
-
- Using bins in the reliability curve, the BS can be decomposed into three parts: reliability, resolution, uncertainty.

$$\text{BS} = \text{REL} - \text{RES} + \text{UNC}$$

3b. Brier score and Brier skill score

- The three components are as follows:

$$\text{REL} = \frac{1}{N} \sum_{k=1}^K N_k (\overline{p_k} - \overline{y_k})^2$$

$$\text{RES} = \frac{1}{N} \sum_{k=1}^K N_k (\overline{y_k} - \overline{y})^2$$

$$\text{UNC} = \overline{y}(1 - \overline{y})$$

- N = total number of examples
- N_k = number of examples in k^{th} bin
- $\overline{p_k}$ = mean predicted probability in k^{th} bin
- $\overline{y_k}$ = event frequency in k^{th} bin
- \overline{y} = event frequency in total dataset

3b. Brier score and Brier skill score

- Define “climatology”: event frequency in total dataset.
- Conceptually:
 - REL = mean squared distance between reliability curve and perfect line
 - RES = mean squared difference between event frequency in bin and in total dataset
 - UNC = irreducible error, a function of climatology only
- REL varies from [0, 1], and the optimal value is 0.
- RES varies from [0, 1], and the optimal value is 1.
- UNC varies from [0, 0.25], and there is no optimal value.
- BS varies from [0, 1], and the optimal value is 0.

$$\text{REL} = \frac{1}{N} \sum_{k=1}^K N_k (\overline{p}_k - \overline{y}_k)^2$$

$$\text{RES} = \frac{1}{N} \sum_{k=1}^K N_k (\overline{y}_k - \overline{y})^2$$

$$\text{UNC} = \overline{y}(1 - \overline{y})$$

3b. Brier score and Brier skill score

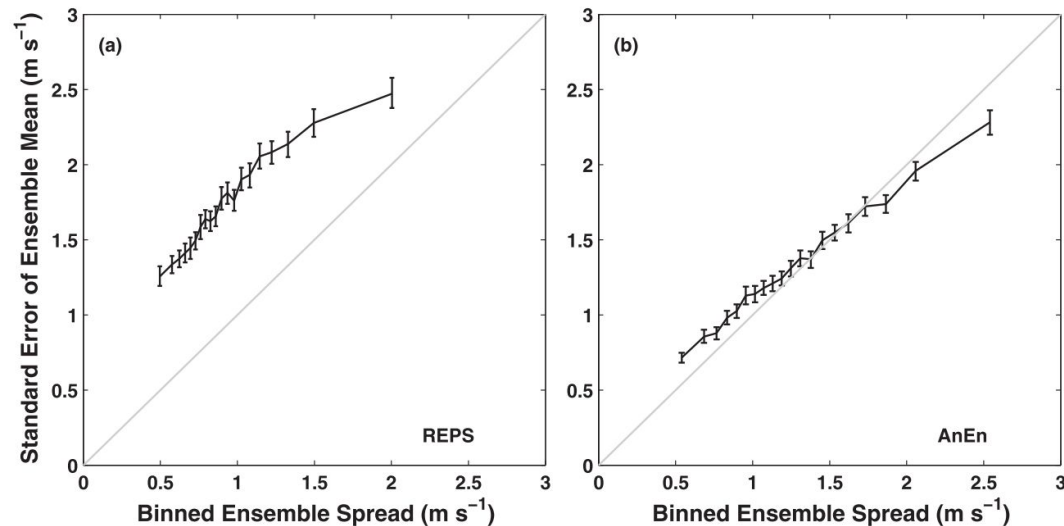
- The Brier skill score (BSS) is defined as:

$$\text{BSS} = \frac{\text{BS}_{\text{ref}} - \text{BS}}{\text{BS}_{\text{ref}}}$$

- BS_{ref} is the BS of a reference model, often a climo model.
- However, the reference model can be anything you want (*e.g.*, the existing state of the art).
- You can interpret BSS values like those of any skill score:
 - Ranges from $(-\infty, 1]$
 - Optimal value is 1
 - Positive value means improvement over baseline

3c. The spread-skill plot

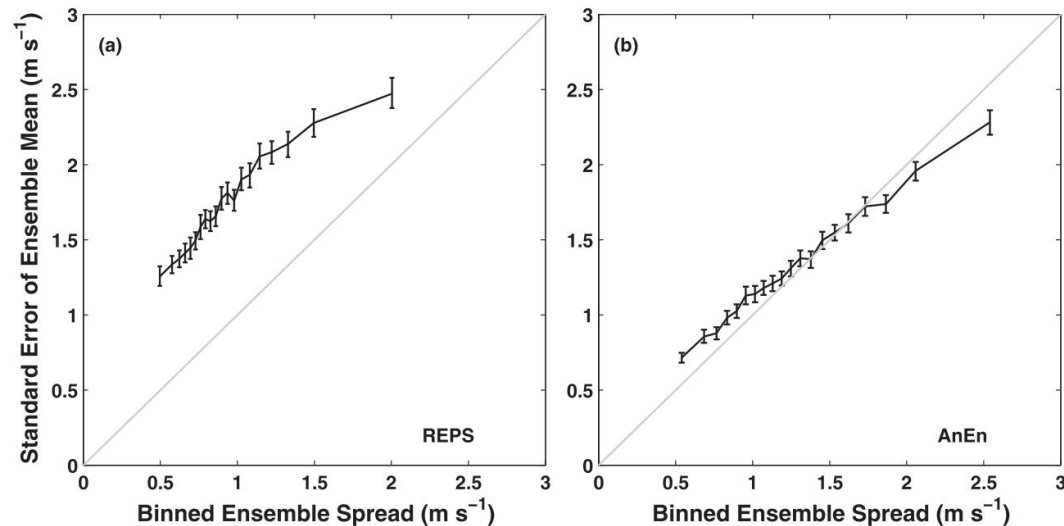
- The spread-skill plot is similar in flavour to the reliability curve.
- However, the spread-skill plot can be used **only** for models that include uncertainty.
- y-coordinate: error (RMSE) of mean model prediction
- x-coordinate: spread (standard deviation) of model predictions
- The standard deviation of model predictions is computed for each example, because there are multiple predictions for each example, because the model includes uncertainty.



Above: example for wind-speed prediction (Figure 6 of [Delle Monache et al. 2013](#))

3c. The spread-skill plot

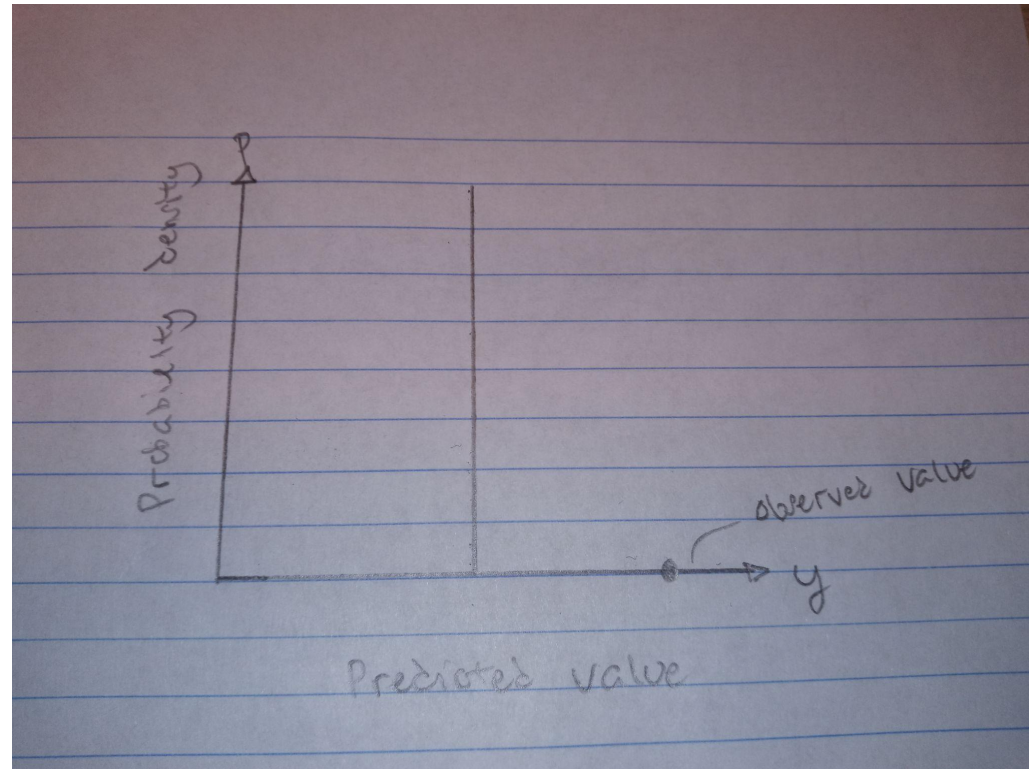
- The spread-skill plot is binned by spread -- in the example shown here, into 20 bins.
- Similar to the reliability curve, which is binned by predicted probability.
- Thus, as in the reliability curve, each point in the spread-skill plot is a mean over all examples in one bin.
- Diagonal grey line: perfect relationship, where spread = skill for all bins.
- Points below grey line: spread is too high, or model is “underconfident,” or model is “overspread”.
- Points above grey line: spread is too low, or model is “overconfident,” or model is “underspread”.



Above: example for wind-speed prediction (Figure 6 of [Delle Monache et al. 2013](#))

3d. Sharpness

- Roughly speaking, sharpness is the opposite of spread.
- Some people diagnose sharpness on its own, without concurrently considering reliability or another measure of skill.
- **Be careful when doing this,** because a model can be very sharp and yet have very little skill.
- The *reductio ad absurdum*: an infinitely sharp model, where the probability distribution of predictions is a step function, yet the observation falls outside the distribution (right).

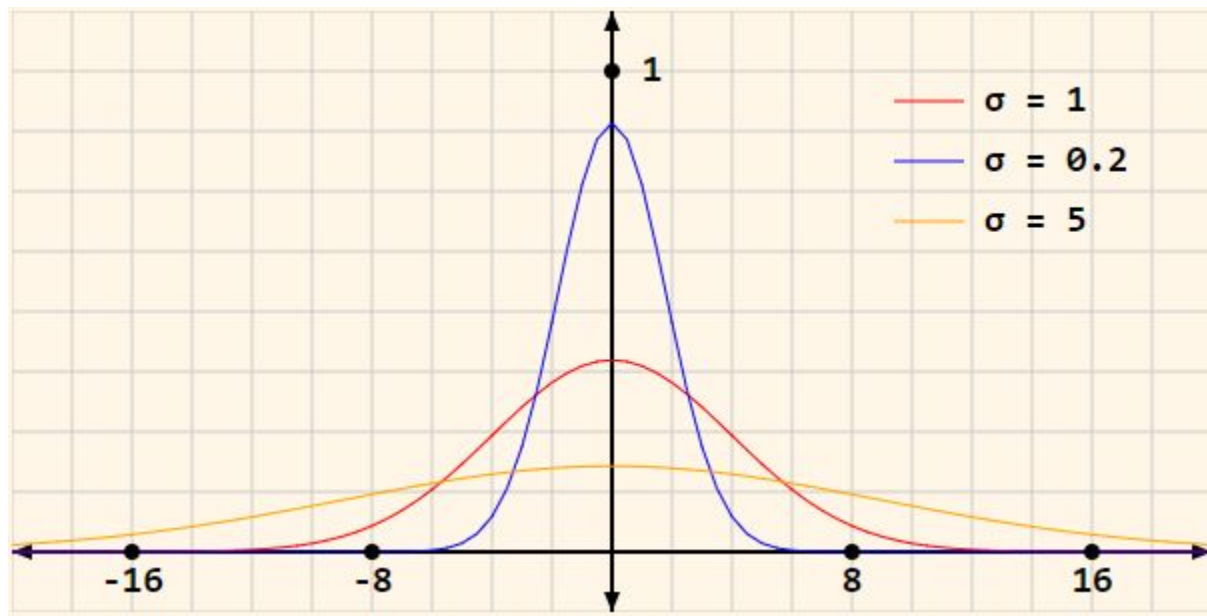


3d. Sharpness

- People typically diagnose sharpness in one of two ways.
 1. Look at predictive histogram, inset in attributes diagram.
 - A sharp model has many forecasts in the highest and lowest bins (*e.g.*, probabilities near 0.0 and 1.0).
 2. Compute mean distance between two percentiles in the predictive PDF (probability-density function).
 - Example: distance between 25th and 75th percentiles.
 - A sharp model has a small mean distance between the two percentiles (*e.g.*, model on previous slide has zero distance between the two percentiles).
 - This distance has the same units as the forecast quantity.
 - Example: if the forecast quantity is wind speed in m s^{-1} , the distance between two percentiles may be 2 m s^{-1} .

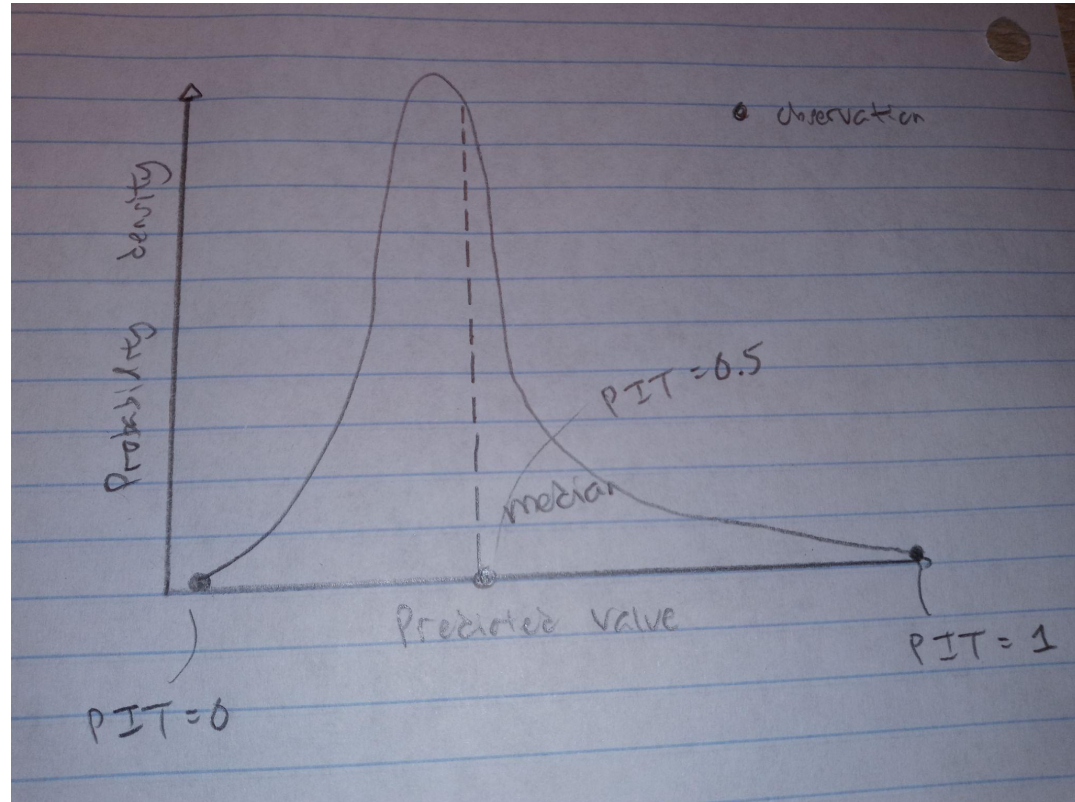
3d. Sharpness

- In the figure below ([image source](#)), the blue PDF is the sharpest and the yellow PDF is the least sharp.



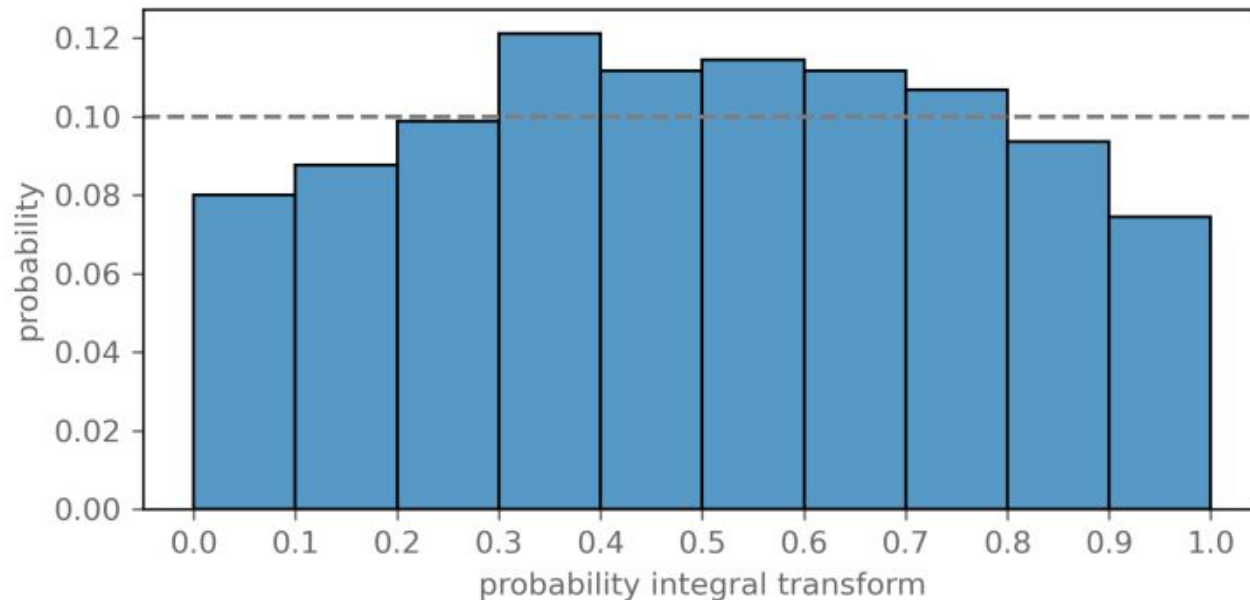
3e. The PIT histogram

- PIT = probability integral transform
- Definition: value that predictive CDF (cumulative density function) attains at observed value
- Alternate definition: quantile of observed value in distribution of predictions
- Examples:
 - Observed value = median of predictive distribution \Rightarrow PIT = 0.5
 - Observed value = max of predictive distribution \Rightarrow PIT = 1.0
 - Observed value = min of predictive distribution \Rightarrow PIT = 0.0



3e. The PIT histogram

- The PIT histogram is a histogram of all PIT values (one for each example).
- For a perfectly calibrated model, the PIT histogram is uniform.
 - In other words, all PIT values occur with the same frequency.
- If the PIT histogram has a hump in the middle, the model has too much spread or is “underconfident” (below; Figure 15 of [Barnes et al. 2021](#)).



- If the PIT histogram has humps on the sides (a lot of values near 0 or 1), the model has too little spread or is “overconfident”.

3e. The PIT histogram

- Atmospheric scientists are typically more familiar with the rank histogram, invented by Talagrand and discussed in [Hamill \(2001\)](#).
- The PIT histogram is a generalization of the rank histogram.
- The rank histogram is used for ensembles, where the ensemble contains a finite number of models and thus generates a finite number of predictions.
- The PIT histogram can be used for any method that generates a predictive distribution, whether the distribution is created by:
 - a) collecting deterministic predictions from each member of an ensemble;
 - b) predicting the quantiles of a distribution;
 - c) predicting the parameters (*e.g.*, mean and standard deviation for Gaussian) of a distribution;
 - d) anything else.
- Happily, the rank histogram and PIT histogram can be interpreted in the same way (uniform = perfectly calibrated; bunched in middle = underconfident; bunched at sides = overconfident).

3f. The continuous ranked probability score

- The continuous ranked probability score (CRPS) is defined as follows:

$$\text{CRPS}(F, y) = \int_{-\infty}^{+\infty} [F(\hat{y}) - H(\hat{y} - y)]^2 d\hat{y}$$

- F = predictive cumulative density function (CDF)
- y = observed value
- \hat{y} = predicted value = variable of integration, ranging from $-\infty$ to $+\infty$
- H = Heaviside step function

- The step function is defined below:

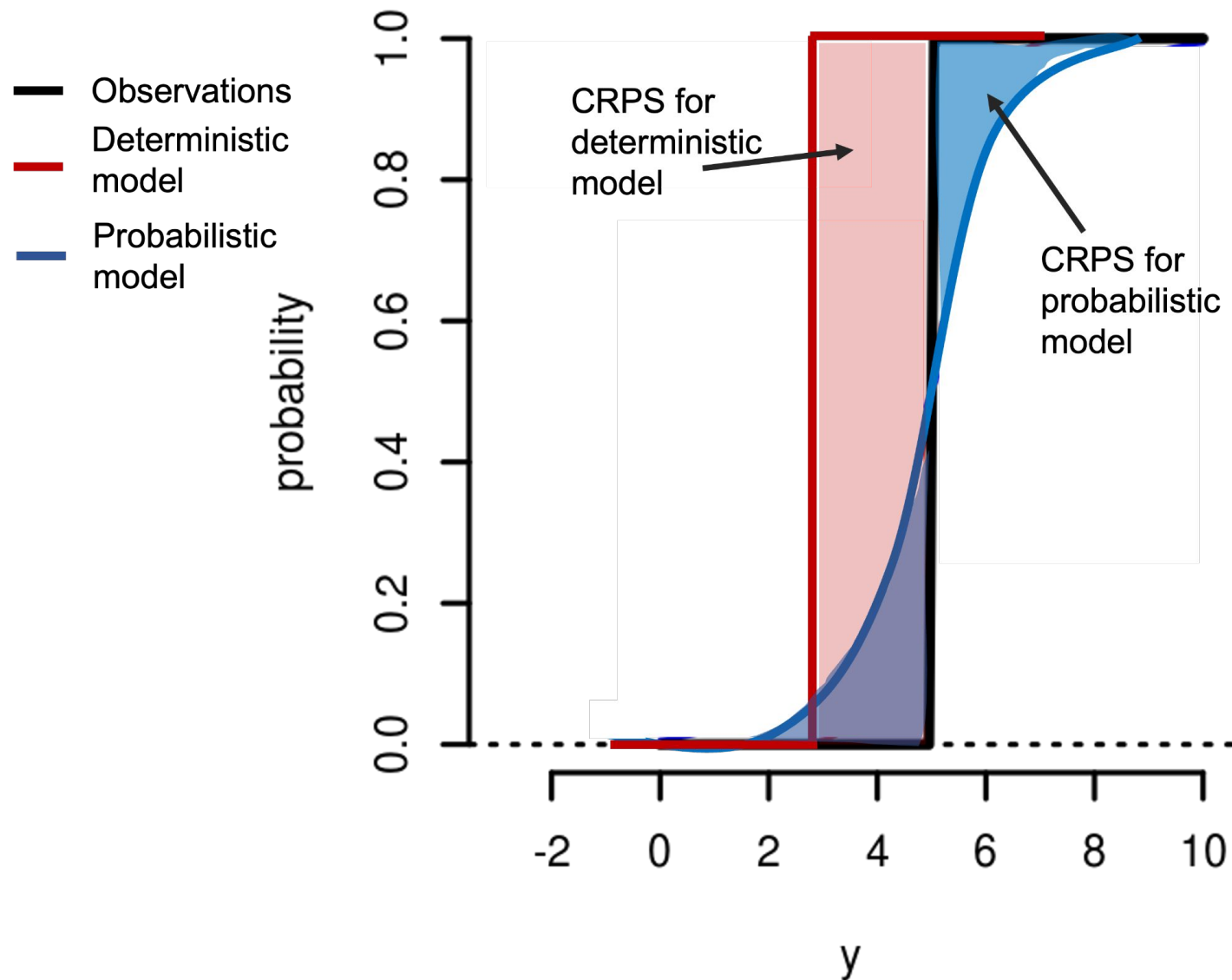
$$H(\hat{y} - y) = \begin{cases} 1, & \hat{y} \geq y \\ 0, & \hat{y} < y \end{cases}$$

- CRPS varies from $[0, \infty)$, and the optimal value is 0.
- To achieve CRPS = 0, one of the two conditions below must be true for every pair of y and \hat{y} :
 - \hat{y} = max of predictive PDF, and $\hat{y} \geq y$
 - \hat{y} = min of predictive PDF, and $\hat{y} < y$

3f. The continuous ranked probability score

- It may not be obvious from the math, but to satisfy these two conditions, a model must be infinitely sharp and perfectly calibrated.
 - In other words, every predictive PDF must be a step function containing the observation.
 - For example, if the observation is 2, the PDF must be a step function with zero density everywhere except 2.
-
- Big advantage of CRPS: it considers the full predictive PDF, not just the mean or standard deviation or certain quantiles or...
 - Some recent work in probabilistic ML has used CRPS as the loss function:
 - [Ghazvinian et al. \(2021\)](#)
 - Will Chapman *et al.* (2021): “Probabilistic predictions from deterministic atmospheric river forecasts with deep learning,” upcoming in *Monthly Weather Review*

3f. The continuous ranked probability score



A few words on literature

Papers on UQ in NNs, in particular Bayesian Neural Networks:

- Be warned: We found most general review papers and tutorials on UQ in NNs to be very hard to understand. Not a good entry point.
- **We found two papers that provide a great entry point (especially the first one):**
 - a. Valentin Jospin, L., Buntine, W., Boussaid, F., Laga, H. and Bennamoun, M.
Hands-on Bayesian Neural Networks - a Tutorial for Deep Learning Users,
arXiv preprint, v2, Sept 2021, <https://arxiv.org/abs/2007.06823>
 - b. Chang, D.T.
Bayesian Neural Networks: Essentials.
arXiv preprint, v1, June 2021, <https://arxiv.org/abs/2106.13594>

4) Methods

- There are many different methods.
- Many are not practical, because computationally too expensive.
- Many are similar.
- **We selected four methods** for this presentation that represent both the most common and the most promising approaches.

The four methods are:

- 1) The analogue ensemble (aka Delle Monache's approach)
- 2) The Barnes approach
- 3) Monte Carlo Dropout (can be interpreted as Bayesian approach) with CPRS loss function
- 4) Bayesian Neural Network - Bayes by Backpropagation (another Bayesian approach)

Method 1: The analogue ensemble

- There are different ways of creating an analogue ensemble (AnEn), but we will focus on [Delle Monache et al. \(2013\)](#).
- The basic idea:
 1. Start with a deterministic NWP model.
 2. Given a new forecast \mathbf{X}_f from the NWP model, find analogues -- *i.e.*, previous forecasts from the same NWP model that are similar to \mathbf{X}_f .
 3. For each analogue forecast, find the *observed* value of the target variable y . Let the observed value corresponding to the i^{th} analogue be $y_i^{(\text{obs})}$.
 4. The AnEn's predictive distribution is created by simply gathering all the $y_i^{(\text{obs})}$.

Method 1: The analogue ensemble

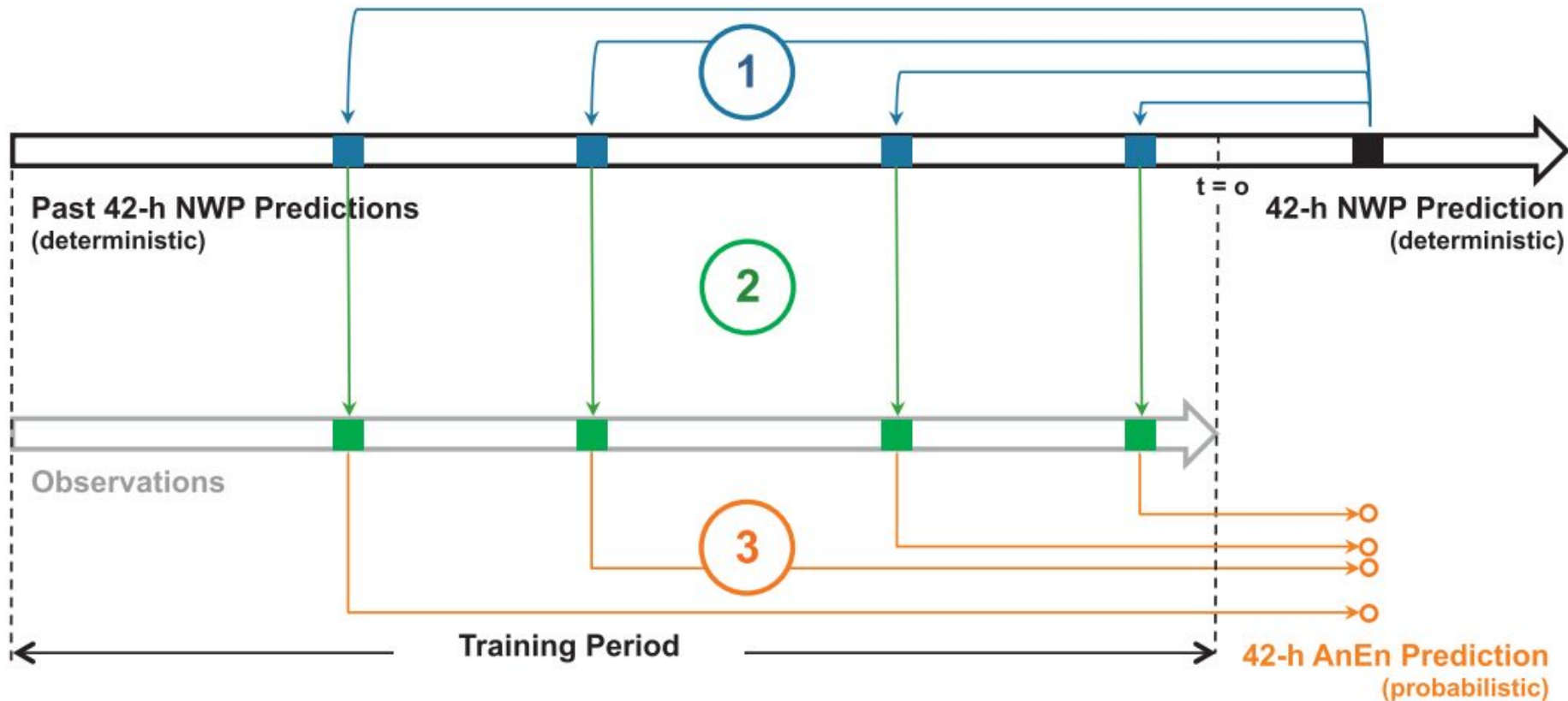


FIG. 3. Schematic representation of the process for finding four members of the analog ensemble (AnEn) at one forecast lead time. A detailed description of the three main steps is found in section 2b(1).

Method 1: The analogue ensemble

- To find analogues (previous forecasts similar to \mathbf{X}_f), you need a distance metric.
- A common choice is the Euclidean distance, or square root of sum of squares of element wise differences.
- For example, if $\mathbf{X}_f = \begin{bmatrix} 10 & 0.7 \\ 12 & 0.6 \end{bmatrix}$ and a previous forecast $\mathbf{X} = \begin{bmatrix} 11 & 0.75 \\ 15 & 0.7 \end{bmatrix}$:

$$d(\mathbf{X}_f, \mathbf{X}) = \sqrt{(10 - 11)^2 + (0.7 - 0.75)^2 + (12 - 15)^2 + (0.6 - 0.7)^2}$$

- Major disadvantage of AnEn: depends on NWP model.
- Major advantages of AnEn:
 - Turns single deterministic NWP model into probabilistic ensemble (other methods for generating probabilities from NWP require a full NWP ensemble)
 - Outperforms other methods for generating probabilities from NWP (see following slides)
- AnEn definitely captures aleatory (data) uncertainty, because it uses past forecasts with similar (but not the same) data.
- We think: AnEn does not capture epistemic (model) uncertainty, because members are created from the same NWP model run with different initial conditions.

Method 1: The analogue ensemble

- Right: rank histograms for REPS (an NWP ensemble) and the AnEn.
- The AnEn clearly has a much better rank histogram, though slightly underconfident.
- The REPS is extremely overconfident, with a large negative bias indicated by the bar at the right side.

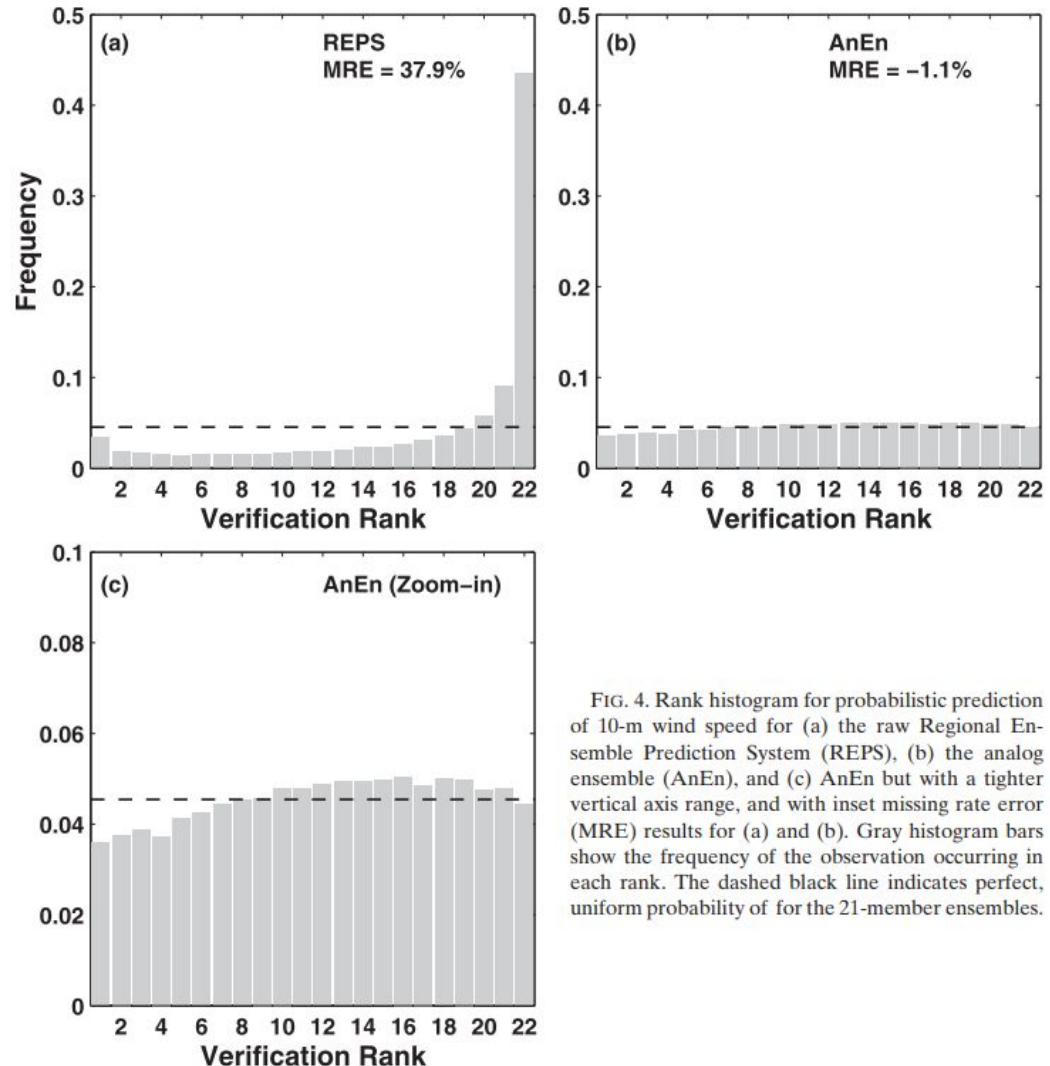


FIG. 4. Rank histogram for probabilistic prediction of 10-m wind speed for (a) the raw Regional Ensemble Prediction System (REPS), (b) the analog ensemble (AnEn), and (c) AnEn but with a tighter vertical axis range, and with inset missing rate error (MRE) results for (a) and (b). Gray histogram bars show the frequency of the observation occurring in each rank. The dashed black line indicates perfect, uniform probability of 1/21 for the 21-member ensembles.

Method 1: The analogue ensemble

- Right: spread-skill plot for REPS (an NWP ensemble) and the AnEn.
- The AnEn is slightly underspread (overconfident) when spread is lower, slightly overspread (underconfident) when spread is higher.
- The REPS is extremely overconfident throughout the domain (*i.e.*, for all values of spread).

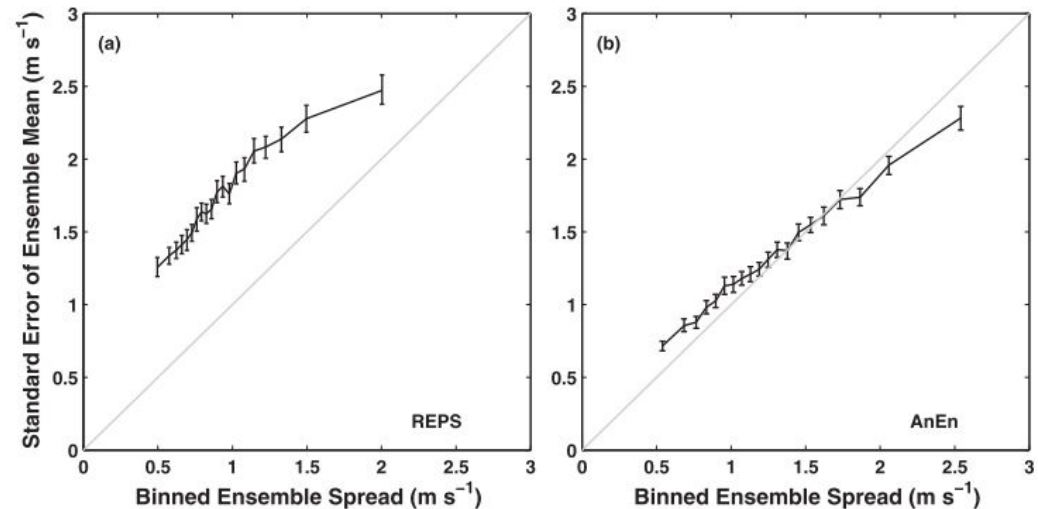


FIG. 6. Binned spread-skill plot for forecast hour 42 of 10-m wind speed for (a) REPS and (b) AnEn. The error bars indicate the 95% bootstrap confidence interval, while the diagonal 1:1 line represents the perfect spread-skill line. For each plot, ensemble spread is binned into 20 equally populated class intervals.

Method 1: The analogue ensemble

- Right: reliability curves for REPS, AnEn, EMOS (ensemble model-output stats), and LR (logistic regression).
- The black curve is the reliability curve; the grey curve is the histogram of predictions.
- The AnEn, EMOS, and LR are all quite reliable; the REPS is very unreliable.
- The REPS has the sharpest histogram (peak for lowest probabilities), but this sharpness comes at the cost of poor reliability.

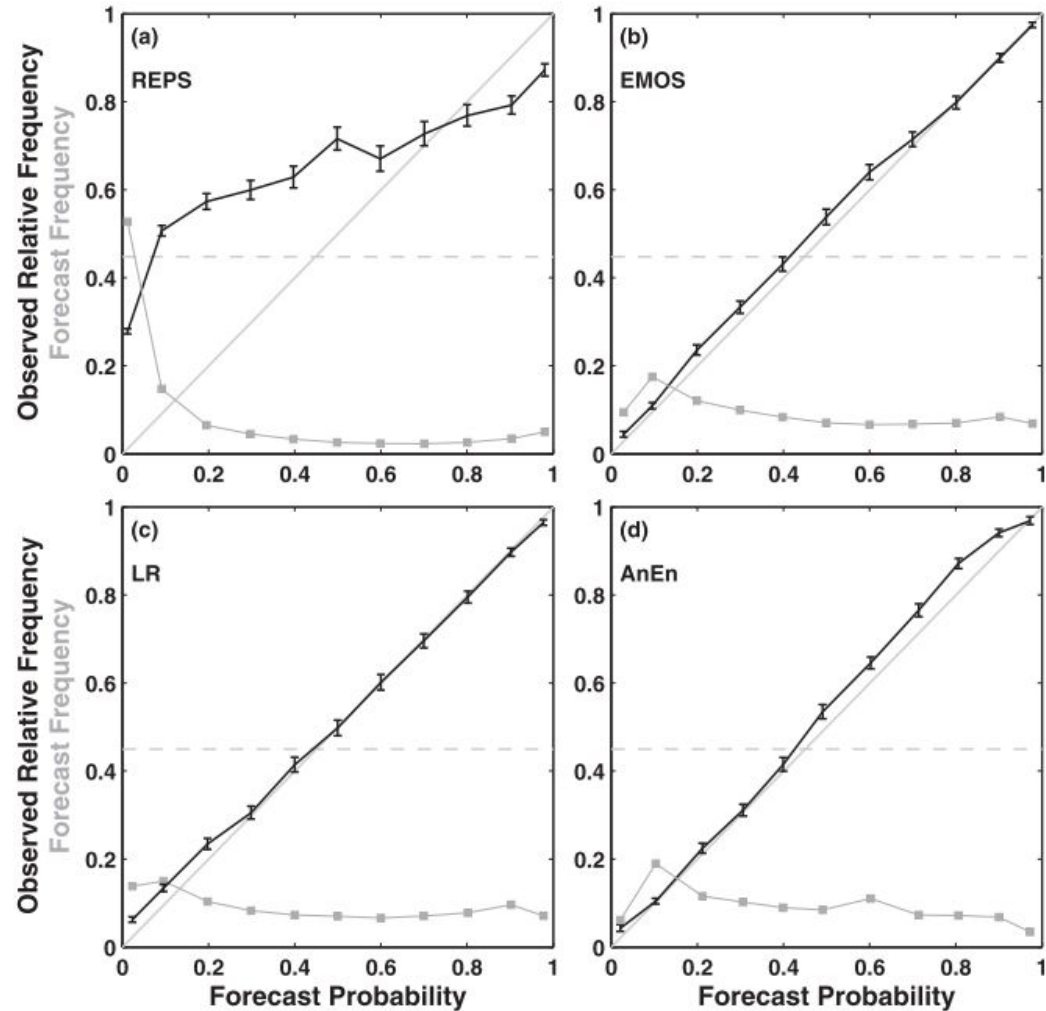


FIG. 7. Reliability (black lines with vertical error bars) and sharpness (gray lines with square marks) for (a) REPS, (b) ensemble model output statistics (EMOS), (c) logistic regression (LR), and (d) AnEn. Results are shown for forecast hour 9 and 10-m wind speed greater than 5 m s^{-1} . The horizontal dashed line represents the event's observed frequency over the verification period (i.e., sample climatology), while the diagonal 1:1 line represents the perfect reliability. The error bars indicate the 95% bootstrap confidence interval.

Method 1: The analogue ensemble

- Right: BSS, resolution, and reliability for the same 4 methods.
- Higher BSS is better; AnEn, EMOS, and LR all much better than REPS.
- Lower reliability is better; AnEn, EMOS, and LR all much better than REPS.
- Higher resolution is better; AnEn, EMOS, and LR all much better than REPS.

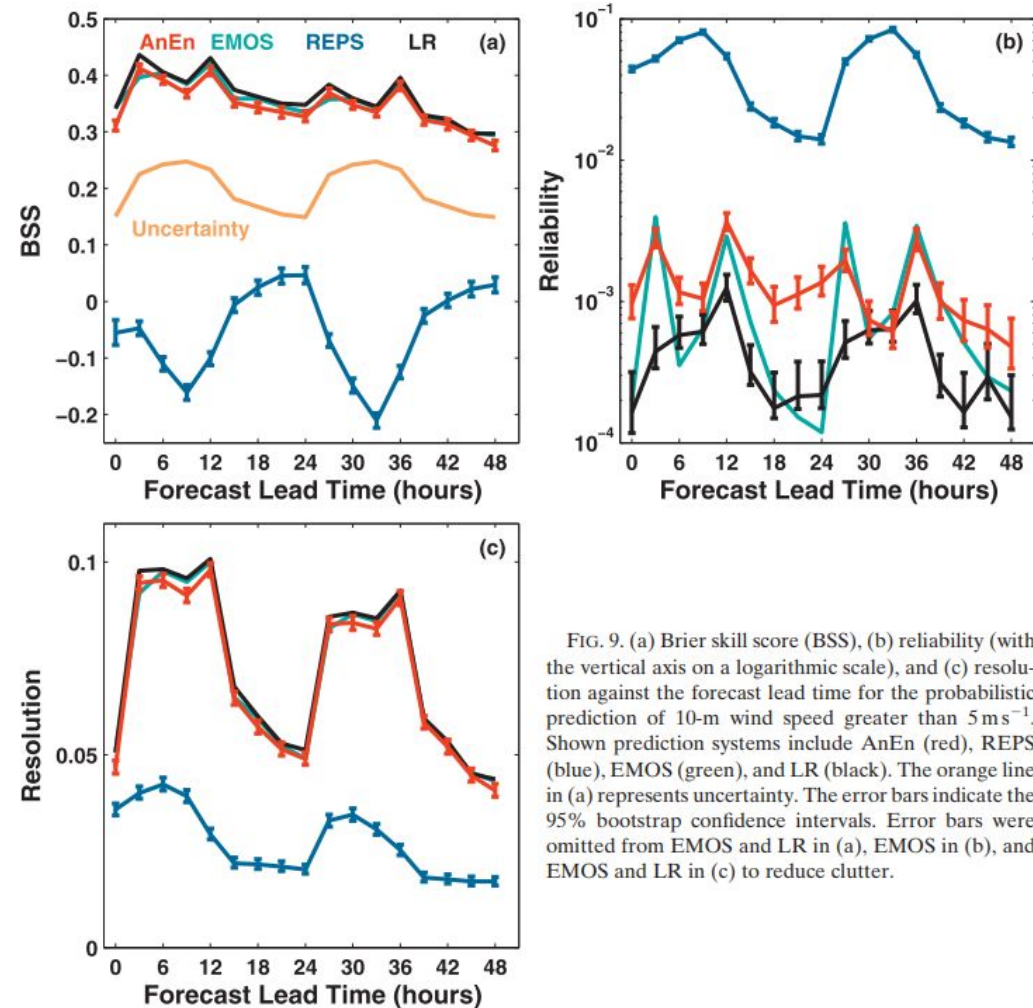


FIG. 9. (a) Brier skill score (BSS), (b) reliability (with the vertical axis on a logarithmic scale), and (c) resolution against the forecast lead time for the probabilistic prediction of 10-m wind speed greater than 5 ms^{-1} . Shown prediction systems include AnEn (red), REPS (blue), EMOS (green), and LR (black). The orange line in (a) represents uncertainty. The error bars indicate the 95% bootstrap confidence intervals. Error bars were omitted from EMOS and LR in (a), EMOS in (b), and EMOS and LR in (c) to reduce clutter.

Method 2: Barnes Approach

Barnes et al. method--Have your neural network predict a probability distribution:

Elizabeth A. Barnes, Randal J. Barnes, Nicolas Gordillo,
Adding Uncertainty to Neural Network Regression Tasks in the Geosciences,

Sept. 2021, <https://arxiv.org/abs/2109.07250>

(this paper gives detailed sample code of how to do this with simple synthetic data!)

- See also related arxiv papers by **Barnes and Barnes** on abstention networks which *use* the above framework
 - Abstention networks for classification (April 2021):
<https://arxiv.org/abs/2104.08281>
 - Abstention networks for regression (April 2021):
<https://arxiv.org/abs/2104.08236>

Method 2: Barnes Approach

Barnes et al. method:

Configure your neural network to output a **probability distribution** instead of simply a single value

You can create a probability distribution with a mean, variance, skewness, and tailweight (related to kurtosis)

You do need to **specify the conditional distribution** ahead of time--Barnes et al. suggest using the *sinh-arcsinh* distribution, as it does not require assumptions of normality, linearity, homoscedasticity, etc.; however, you could also specify a distribution such as lognormal, gamma, or beta

Method 2: Barnes Approach

Barnes et al. method:

Keep in mind that a histogram may not capture the full variability of your data

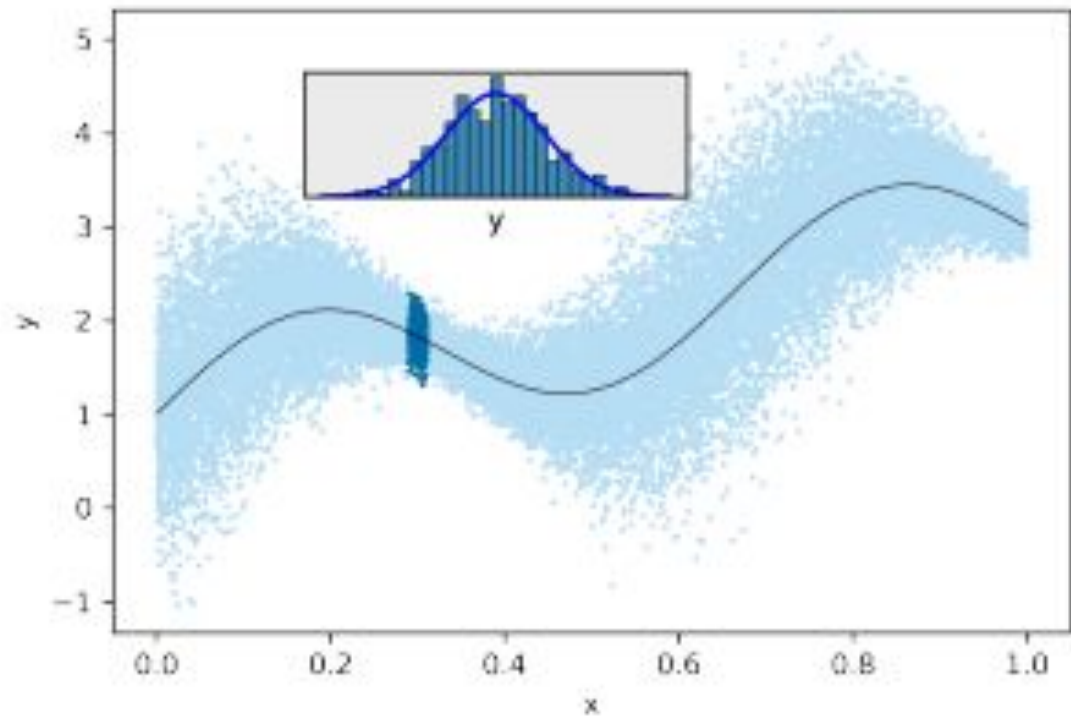
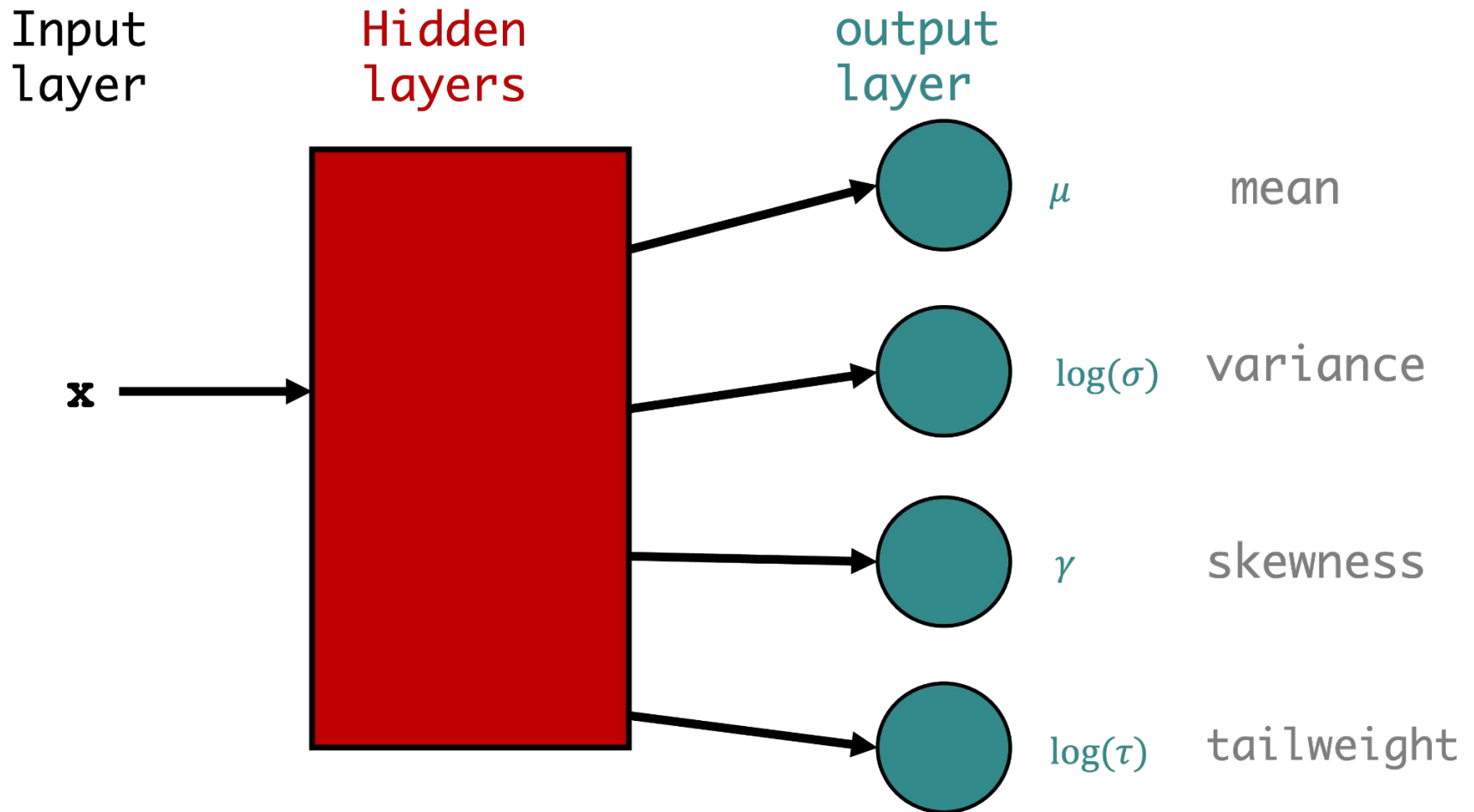
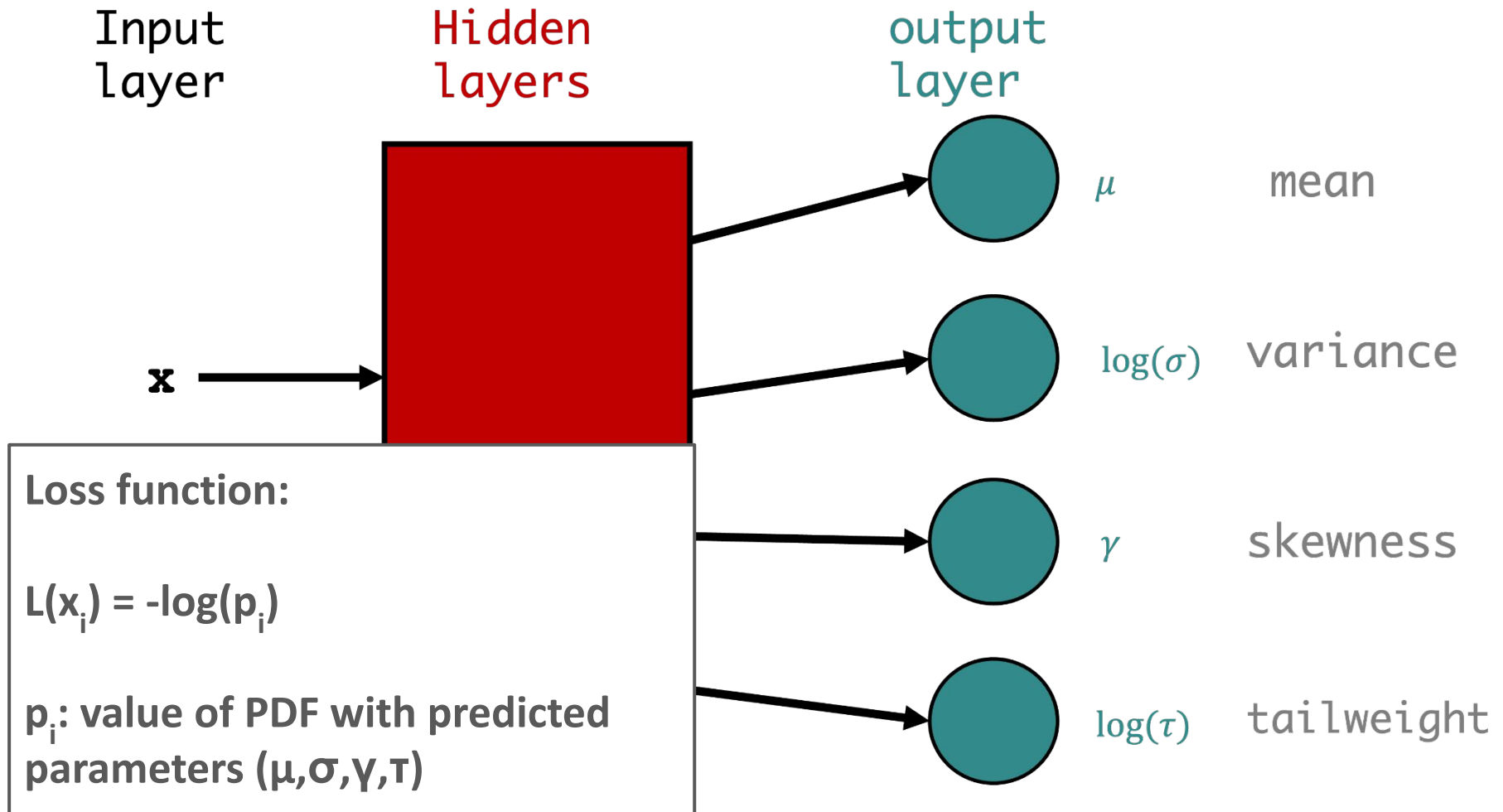


Figure 5 from
Barnes et al. 2021

Method 2: Barnes Approach



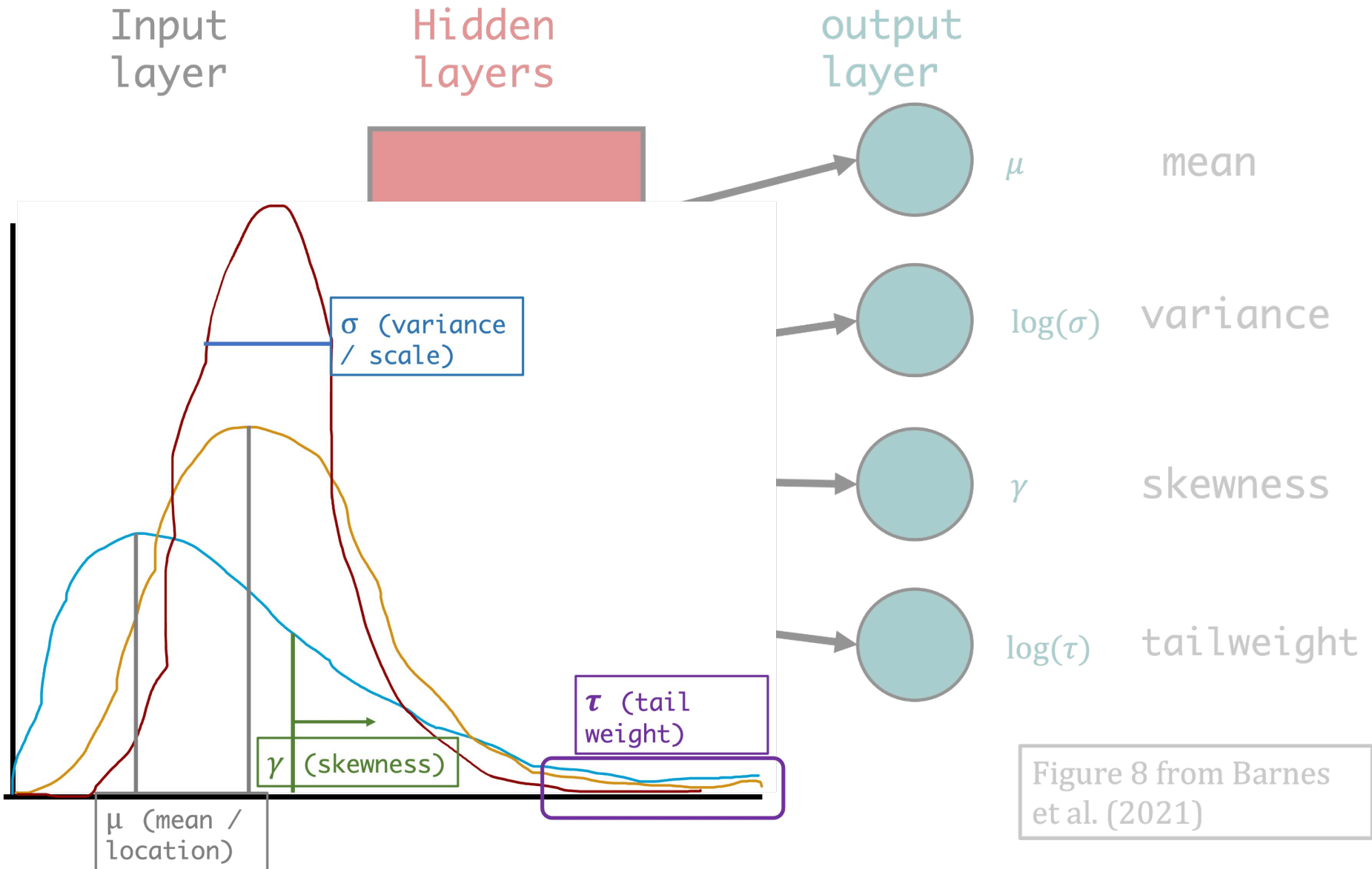
Method 2: Barnes Approach



<https://arxiv.org/abs/2109.07250>

Figure 8 from Barnes et al. (2021)

Method 2: Barnes Approach



ML core meeting
Nov 10, 2021

Uncertainty Quantification Methods for Neural Networks (Part 2)

Ryan Lagerquist
Marie McGraw
Imme Ebert-Uphoff

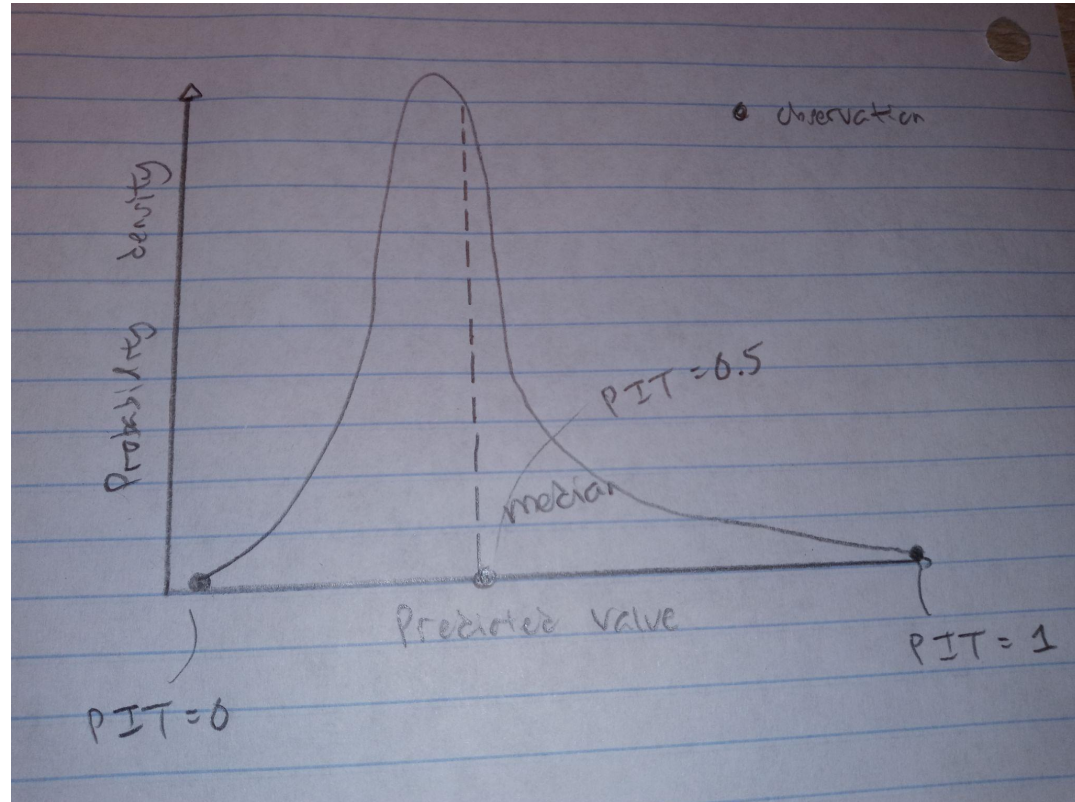
First: What we skipped last time

One of the evaluation methods skipped last time:

The PIT histogram.

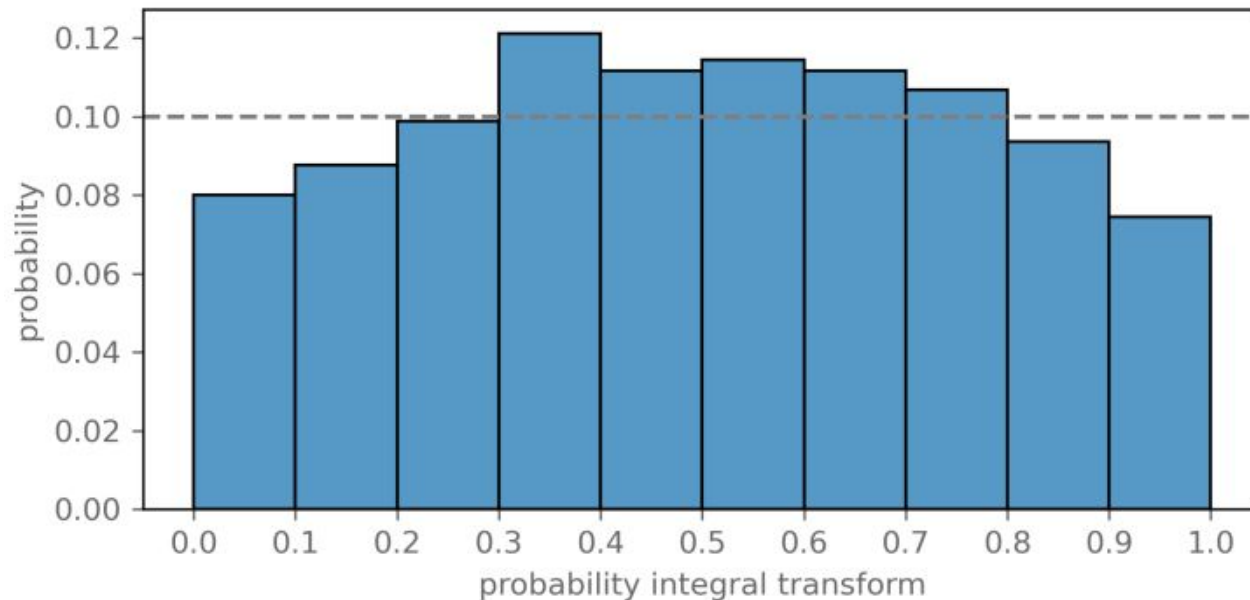
3e. The PIT histogram

- PIT = probability integral transform
- Definition: value that predictive CDF (cumulative density function) attains at observed value
- Alternate definition: quantile of observed value in distribution of predictions
- Examples:
 - Observed value = median of predictive distribution \Rightarrow PIT = 0.5
 - Observed value = max of predictive distribution \Rightarrow PIT = 1.0
 - Observed value = min of predictive distribution \Rightarrow PIT = 0.0



3e. The PIT histogram

- The PIT histogram is a histogram of all PIT values (one for each example).
- For a perfectly calibrated model, the PIT histogram is uniform.
 - In other words, all PIT values occur with the same frequency.
- If the PIT histogram has a hump in the middle, the model has too much spread or is “underconfident” (below; Figure 15 of [Barnes et al. 2021](#)).



- If the PIT histogram has humps on the sides (a lot of values near 0 or 1), the model has too little spread or is “overconfident”.

3e. The PIT histogram

- Atmospheric scientists are typically more familiar with the rank histogram, invented by Talagrand and discussed in [Hamill \(2001\)](#).
- The PIT histogram is a generalization of the rank histogram.
- The rank histogram is used for ensembles, where the ensemble contains a finite number of models and thus generates a finite number of predictions.
- The PIT histogram can be used for any method that generates a predictive distribution, whether the distribution is created by:
 - a) collecting deterministic predictions from each member of an ensemble;
 - b) predicting the quantiles of a distribution;
 - c) predicting the parameters (*e.g.*, mean and standard deviation for Gaussian) of a distribution;
 - d) anything else.
- Happily, the rank histogram and PIT histogram can be interpreted in the same way (uniform = perfectly calibrated; bunched in middle = underconfident; bunched at sides = overconfident).

UQ Methods for NNs

Discussed last time:

Method 1: The analogue ensemble (Delle Monache et al.)

Method 2: Barnes approach

Non-Bayesian methods



Today:

Method 3: MC Dropout with CRPS loss

Method 4: Bayesian Neural Networks

Bayesian methods



Methods 3&4 are both based on Bayesian inference.

Method 3: MC Dropout with CRPS loss

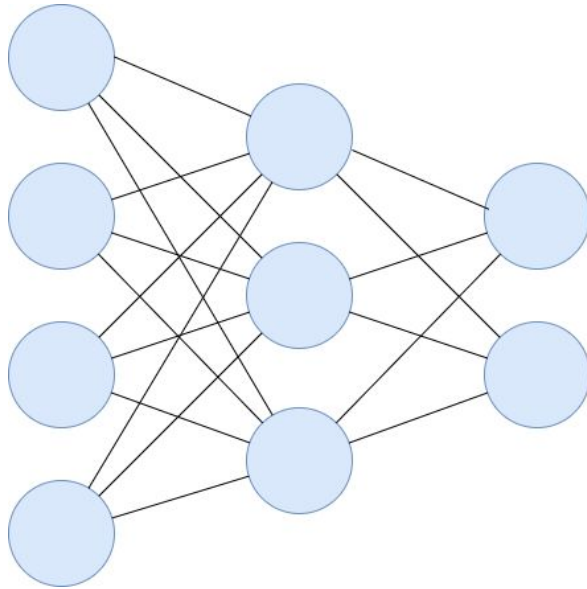
Literature on Monte Carlo Dropout:

- **Original paper:** Gal, Y. and Ghahramani, Z., 2016, June. **Dropout as a bayesian approximation: Representing model uncertainty in deep learning**. In international conference on machine learning (pp. 1050-1059). PMLR.
<http://proceedings.mlr.press/v48/gal16.pdf>
- **Quick summary:** Michal Oleszak, **Monte Carlo Dropout**, *Towards data science*, Sep 20, 2020. <https://towardsdatascience.com/monte-carlo-dropout-7fd52f8b6571>

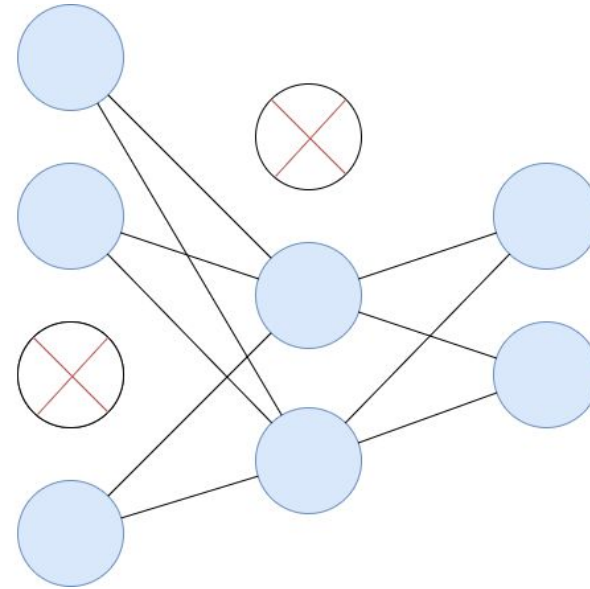
Key command in Keras:

```
model.add( keras.layers.Dropout( dropout_rate ) )
```

Method 3: Dropout



Regular Neural Network



Same network with two neurons dropped out
(reassigned for each batch)

Image source: Michal Oleszak, **Monte Carlo Dropout**, *Towards data science*, Sep 20, 2020.
<https://towardsdatascience.com/monte-carlo-dropout-7fd52f8b6571>

Method 3: Monte Carlo Dropout

Standard use of dropout: Dropout during training - to avoid overfitting

- Dropout is *usually* used only during NN training to avoid overfitting:
- Idea: during training randomly ignore neurons according to specified drop out probability:
 - neurons with drop-out rate=0 → never dropped
 - neurons with drop-out rate=0.5 → dropped about 50% of the time.
 - **Dropout rate** becomes an additional hyperparameter
- For each batch: decide which neurons are dropped. Different neurons active for each batch!
- NN is forced to learn to distribute signals across many neurons (redundancy), because it cannot rely on any neuron to be connected.
- Since different neurons are dropped in each batch, we effectively create an **ensemble**

Different use - to obtain uncertainties: Drop-out during prediction (after having trained with dropout)

- Dropout can be interpreted as Bayesian approximation of Gaussian process.
- Idea: Use dropout when running the model to generate predictions
 - each dropout version provides a different NN model
 - provides ensemble of NN models
 - ensemble of predictions → can get uncertainty estimate from that ensemble.
- Loss function: Best coupled with using special loss function during training, see next slide.
- *It was shown that MC Dropout can be interpreted as a special case of Bayesian inference - although originally it was not derived as such.*

Michał Oleszak, **Monte Carlo Dropout**, *Towards data science*, Sep 20, 2020.

<https://towardsdatascience.com/monte-carlo-dropout-7fd52f8b6571>

Method 3: Monte Carlo Drop-Out

Probabilistic predictions require you to train your model using probabilistic-friendly loss functions

One option: The CRPS--essentially, an analogue of mean absolute error (MAE) for probabilistic forecasts

$$\text{CRPS}(F, y) = \int_{-\infty}^{+\infty} [F(\hat{y}) - H(\hat{y} - y)]^2 d\hat{y}$$

1 or 0, depending

prediction

observations

Method 3: Monte Carlo Drop-Out

Probabilistic predictions require you to train your model using probabilistic-friendly loss functions

One option: The CRPS--essentially, an analogue of mean absolute error (MAE) for probabilistic forecasts

$$\text{CRPS}(F, y) = \int_{-\infty}^{+\infty} [F(\hat{y}) - H(\hat{y} - y)]^2 d\hat{y}$$

1 or 0, depending

prediction

observations

Classification problems using a similar approach discussed in [Scheuerer et al. \(2020\)](#).

Method 4: Bayesian Neural Networks (BNNs)

Introductory references (repeated from before):

- Valentin Jospin, L., Buntine, W., Boussaid, F., Laga, H. and Bennamoun, M.
Hands-on Bayesian Neural Networks - a Tutorial for Deep Learning Users,
arXiv preprint, v2, Sept 2021, <https://arxiv.org/abs/2007.06823>
- Chang, D.T.
Bayesian Neural Networks: Essentials.
arXiv preprint, v1, June 2021, <https://arxiv.org/abs/2106.13594>

Additional references, **especially regarding implementation in TensorFlow**

- Tran, D., Dusenberry, M.W., van der Wilk, M. and Hafner, D.
Bayesian layers: A module for neural network uncertainty.
arXiv preprint, Dec 2018, <https://arxiv.org/abs/1812.03973>

Implementation:

On Github: <https://github.com/tensorflow/tensor2tensor>

- TensorFlow probability: <https://www.tensorflow.org/probability>

Specifics on Edward2
framework, which is used
in TensorFlow Probability.

Bayesian Neural Network

Standard (deterministic) NN:

- Weights & biases are parameters to be learned;
- Activation functions are fixed (pre-selected).

Bayesian Neural Networks

= Neural Networks where

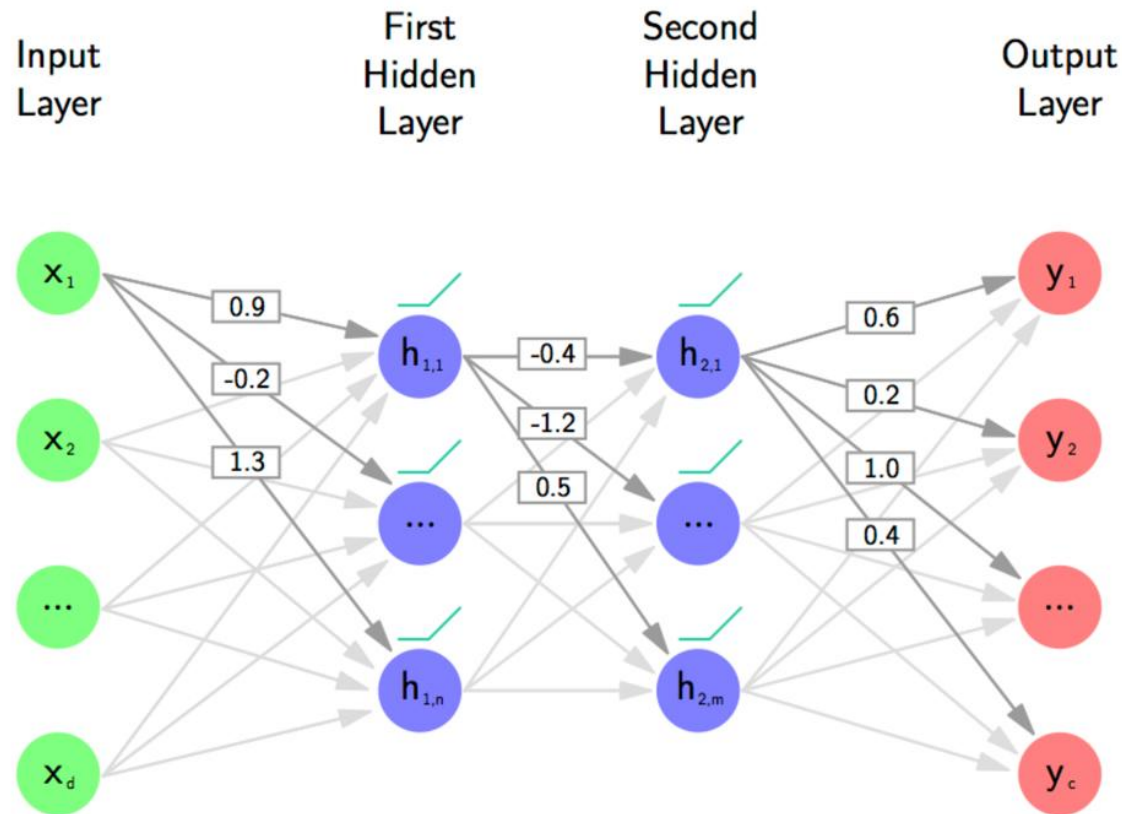
- either weights & biases,
- or activation functions

in the layers are probabilistic.

Most common BNN type - **we will only focus on this type here:**

- **Activation functions are fixed;**
- **Weights and biases are modeled as probabilistic.**

Bayesian Neural Network



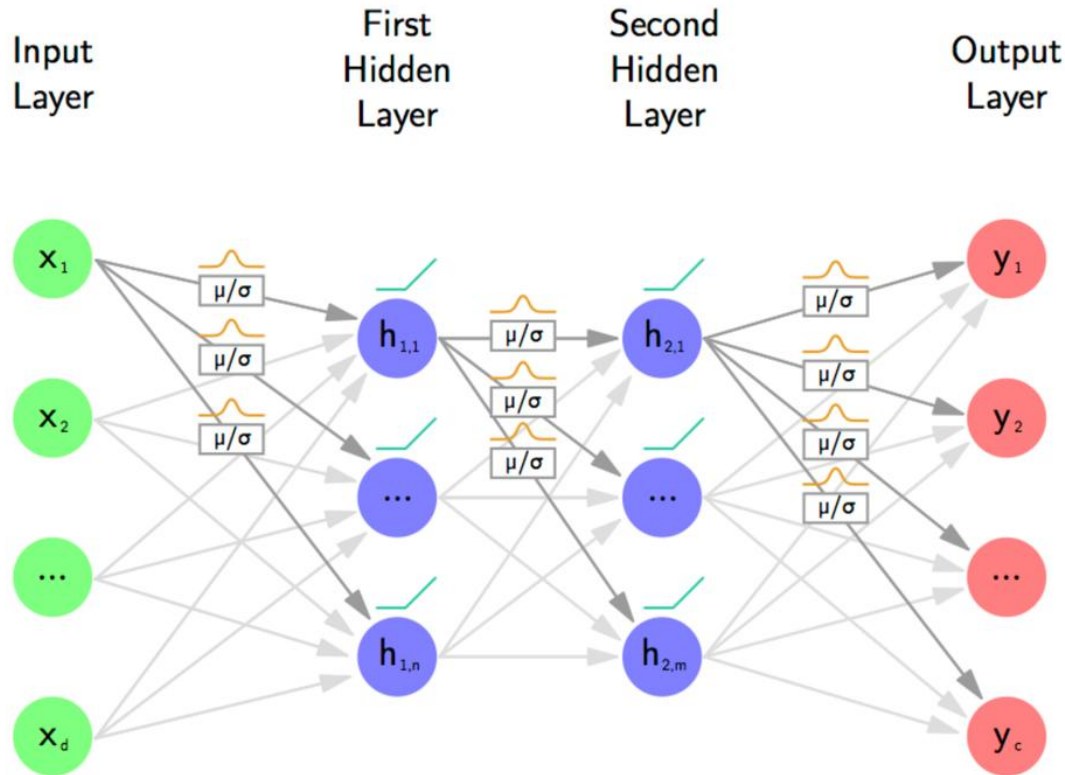
Standard (Deterministic) Neural Network:

- all weights and biases are scalars to be learned

Image credit: Gluon Educational Resources, Chapter 18 on Variational methods and uncertainty.

https://gluon.mxnet.io/chapter18_variational-methods-and-uncertainty/bayes-by-backprop.html

Bayesian Neural Network



Bayesian Neural Network:

- **Probabilistic layers:** all weights and biases are probability distributions to be learned.
- Shown here: Gaussian distribution (μ, σ) assumed for each weight, but does not have to be Gaussian.

Image credit: Gluon Educational Resources, Chapter 18 on Variational methods and uncertainty.

https://gluon.mxnet.io/chapter18_variational-methods-and-uncertainty/bayes-by-backprop.html

Bayesian Neural Network (BNN)

BNN type considered here:

- Weights and biases are modeled as probabilistic distributions.

How do we train such a model?

Can't just use standard back propagation for probabilistic layers.

→ Use Bayesian inference instead!

→ Thus the name “Bayesian Neural Network”.

Two common Bayesian learning approaches for inference:

1. **Markov Chain Monte Carlo** (approximate by sampling):
Not practical for most applications - computationally too expensive.
2. **Variational Inference** (approximate by optimization):
Much more feasible. Leads to several methods. Considered here.

Bayesian Neural Network (BNN)

At first sight - this might still seem like an impossible task!

Variational inference learns posterior distributions from prior distributions.

→ Key challenges:

- 1) How do you identify the **prior distributions for the NN weights**?
- 2) How do you implement **Bayesian inference efficiently in a NN framework to calculate the posterior distribution for all the NN weights**?

The solution:

- Can't do exact Bayesian learning.
- Use lots of simplifying assumptions!
- **Approximations** seem to work pretty well.
- Next slides.

Bayesian Neural Network (BNN)

Challenge 1: How to define prior distribution for weight parameters?

- For variational inference, we need to “know” prior distribution of weights and biases.
- Turns out - that problem is not even well defined when using several probabilistic layers. Lots of redundancy - no unique solution exists!

Solution for Challenge 1:

- Common practice: **Just assume prior distribution for each parameter to be Gaussian!**
- Not particularly good and not particularly bad. But simple + easy. Good enough.
Note: Assuming Gaussian distribution corresponds to using regularization.
- Note:
 - There are also some approaches that *learn* the prior distribution.
 - But most approaches just assume Gaussian distribution.

Bayesian Neural Network (BNN)

Challenge 2: How to implement Variational Inference efficiently?

Solution: Variational inference can be implemented as

1. Probabilistic back propagation
2. Bayes-by-backprop

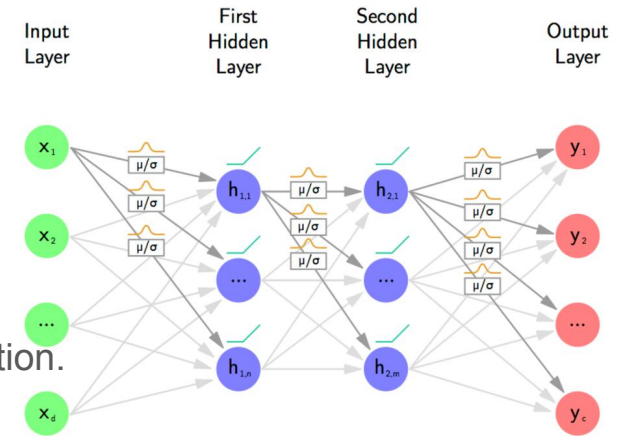
Bayes-by-backprop tends to be more common.

Bayes-by-backprop - Basic idea:

- Assume certain distribution for each weight, e.g., normal distribution.
- Instead of learning ONE scalar for each weight, we learn TWO scalars (μ and σ) for each weight.
- Choose special loss function - typically has two parts:
 - Part 1: fitting the data
 - Part 2: regularization term that penalizes distance between prior and posterior distribution.
- Still needs some tricks to make back propagation work.

But with those tricks we can then apply standard gradient descent method to train our probabilistic network.

For details, see Jospin et al. (<https://arxiv.org/abs/2007.06823>).



Result:

- Transformed our variational inference task into NN framework (with a few modifications).
- We can now implement Bayesian inference using Gradient descent methods of NNs!
- TensorFlow Probability can handle all of that for us.

Bayesian Neural Network (BNN)

Challenge 3: Even with these simplifications, **if more than just a few layers are probabilistic most research group report that they quickly run out of memory!**

Common solution:

- Having all layers be probabilistic is highly redundant anyway.
- Simplify:
 - Keep most layers deterministic,
 - only make a few layers probabilistic.
- That seems to be sufficient for most applications - according to some papers.
- Best practices:
 - Making just 1-2 layers probabilistic seems to work well.
 - Using the last layers (closest to output) seems to work best.

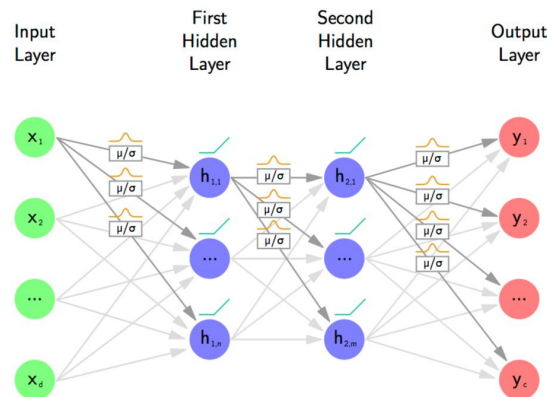
But note:

- Orescanin et al. succeeded in making ***all*** layers of deep NN probabilistic without running out of memory for remote sensing application! See discussion of that paper later.
- Some papers claim that UQ estimate is better when all layers are probabilistic.
- But that also requires **a lot more training data**, so might not always be feasible for us.

Bayesian Neural Network (BNN)

As mentioned before:

- We don't have to implement probabilistic layers and variational inference ourselves!
- TensorFlow's probability (TFP) library provides variety of probabilistic layers.
- Replace a few deterministic layers with probabilistic layers (lots of additional parameters to be specified), choose suitable loss function, and you got a probabilistic NN.
- In practice: probably many questions still to be answered on best use of these methods for CIRA applications.
- **We (Marie, Ryan, Imme) are planning to explore this approach in the coming months for CIRA applications. (Planning to collaborate with Marko Orescanin, see his paper later.)**



Applications

On the following slides:

- Selected papers from weather/climate related applications that use these UQ methods for NNs.
- Not an exhaustive list - I'm sure we missed some.

Group 1: Analogue Ensemble (Delle Monache et al.)

Delle Monache, L., Eckel, F.A., Rife, D.L., Nagarajan, B. and **Searight, K.**, 2013. Probabilistic weather prediction with an analog ensemble. Monthly Weather Review, 141(10), pp.3498-3516.

Our CIRA
colleague in
Boulder!

[Already discussed in Part 1 of this presentation (Oct 13).]

Method used:

- The analogue ensemble (AnEn) turns a single deterministic NWP model into an ensemble.
- For target variable y and valid time t , the AnEn takes the NWP forecast valid at t , finds times $\{t^*\}$ with similar past forecasts from the same NWP model, then finds values of y at times $\{t^*\}$.
- Values of y at times $\{t^*\}$ form the probability distribution.

Application: Predicting 10-m wind speed and 2-m temperature at various weather stations across the U.S.

Key comments:

- Computationally efficient -- creates NWP ensemble from single NWP model.
- Limited -- requires long history of forecasts from the same NWP model with the same configuration, *i.e.*, a reforecast.

Group 2: Predicting Probability Distributions (Barnes group)

Barnes, E.A., Barnes, R.J. and Gordillo, N., 2021. Adding Uncertainty to Neural Network Regression Tasks in the Geosciences. arXiv preprint arXiv:2109.07250.

(*related*) Barnes, E.A. and Barnes, R.J., 2021. Controlled abstention neural networks for identifying skillful predictions for regression problems. arXiv preprint arXiv:2104.08236.

[Already discussed in Part 1 of this presentation (Oct 13).]

Method used: Neural network estimates **probability distribution** rather than a single value. Outputs correspond to mean, variance, skewness, and tailweight of distribution

Application: Predict conditional distribution of SST anomalies (using synthetic data from a benchmark dataset). Also examples using simple 1-D data.

Key comments:

- **Flexible:** Only the output layer and loss function are modified, so it does not require any specific network architecture

Group 2: Predicting Probability Distributions (Barnes group)

Method used: Neural network estimates **probability distribution** rather than a single value. Outputs correspond to mean, variance, skewness, and tailweight of distribution

Application: Predict conditional distribution of SST anomalies (using synthetic data from a benchmark dataset). Also examples using simple 1-D data.

Key comments:

- **Adaptable:** Only the output layer and loss function are modified, so this approach does not require any specific network architecture
- **Flexible:** This approach does not require any assumptions about linearity, homoscedasticity, or normality
- You do **NOT** need to select a prior beforehand (unlike Bayesian methods)
- You **DO** need to select the distribution (that you are drawing from to describe your data) ahead of time--the authors suggest the *sinh-arcsinh* distribution, but depending on application other choices (i.e., gamma) may be preferable
- Easy to implement in TensorFlow (and authors provide sample code)

Group 3: Bayesian Neural Networks (Orescanin et al.)

Orescanin, M., Petković, V., Powell, S.W., Marsh, B.R. and Heslin, S.C., 2021. Bayesian Deep Learning for Passive Microwave Precipitation Type Detection. IEEE Geoscience and Remote Sensing Letters.

Method used:

- Bayesian neural network - making all NN weights probabilistic.

Application:

- Classify precipitation type (stratiform/convective) based on passive MW imagery
- Namely, map from raw GMI data to precipitation type (stratiform/convective)
- Goal: provide two outputs:
 - i. Map of precipitation type: indicates stratiform/convective per pixel.
 - ii. Map of uncertainty: indicates how much to trust classification per pixel.

Set-up:

- Global precipitation measurement (GPM) carries:
 - passive Microwave Imager (GMI)
 - dual-frequency precipitation radar (DPR)
- Use DPR to assign a label to each GMI pixel
- Each sample contains:
 - Patches of (9x25) pixels of GMI data (125km x 125km area).
 - For each pixel: feature vector of 13 Tbs (GMI)
 - Label: Binary pixel labels obtained from DPR.
- 14 million samples available for training and testing.

Group 3: Bayesian Neural Networks (Orescanin et al.)

Approach:

- NN architecture: CNN of type ResNet
- **Baseline model: deterministic ResNet**
Comment: ResNet is a deep network with lots of parameters.
- **New model: probabilistic ResNet**
 - Turn *all* layers of ResNet into Bayesian layers:
 - all weights of all layers are modeled as Gaussian distribution.
 - That doubles # of parameters:
each weight, w , is replaced by two parameters: (μ, σ)
- **Key comments:**
 - Implementation: Implemented in TensorFlow probability.
 - Impressive implementation: they manage to turn *all* layers of ResNet into Bayesian layers, without running out of memory!
 - But still needs lots of data (they have 14M samples!)

Group 3: Bayesian Neural Networks (Orescanin et al.)

Results:

high entropy =
high uncertainty

- **Deterministic ResNet**: 86% accuracy, no uncertainty estimate.
- **Probabilistic ResNet**: 90% accuracy, and yields uncertainty estimate.

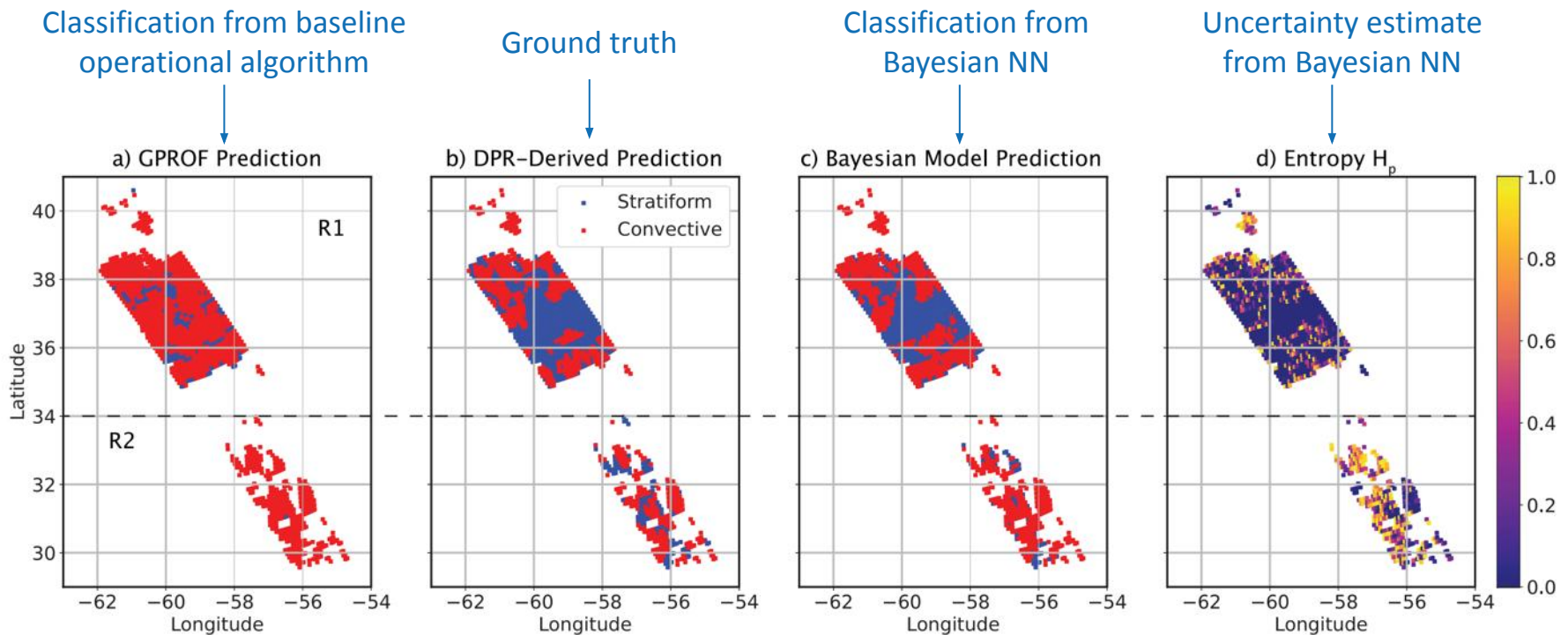


Fig. 1. Case study-August 11, 2018, GPM orbit 025293. From left to right: (a) GPROF prediction: a current operational benchmark. (b) DPR-derived product: the truth/label. (c) Bayesian model prediction ResNetV2. (d) Predictive H_p map (uncertainty map) for the Bayesian model prediction.

Group 3: Bayesian Neural Networks (Orescanin et al.)

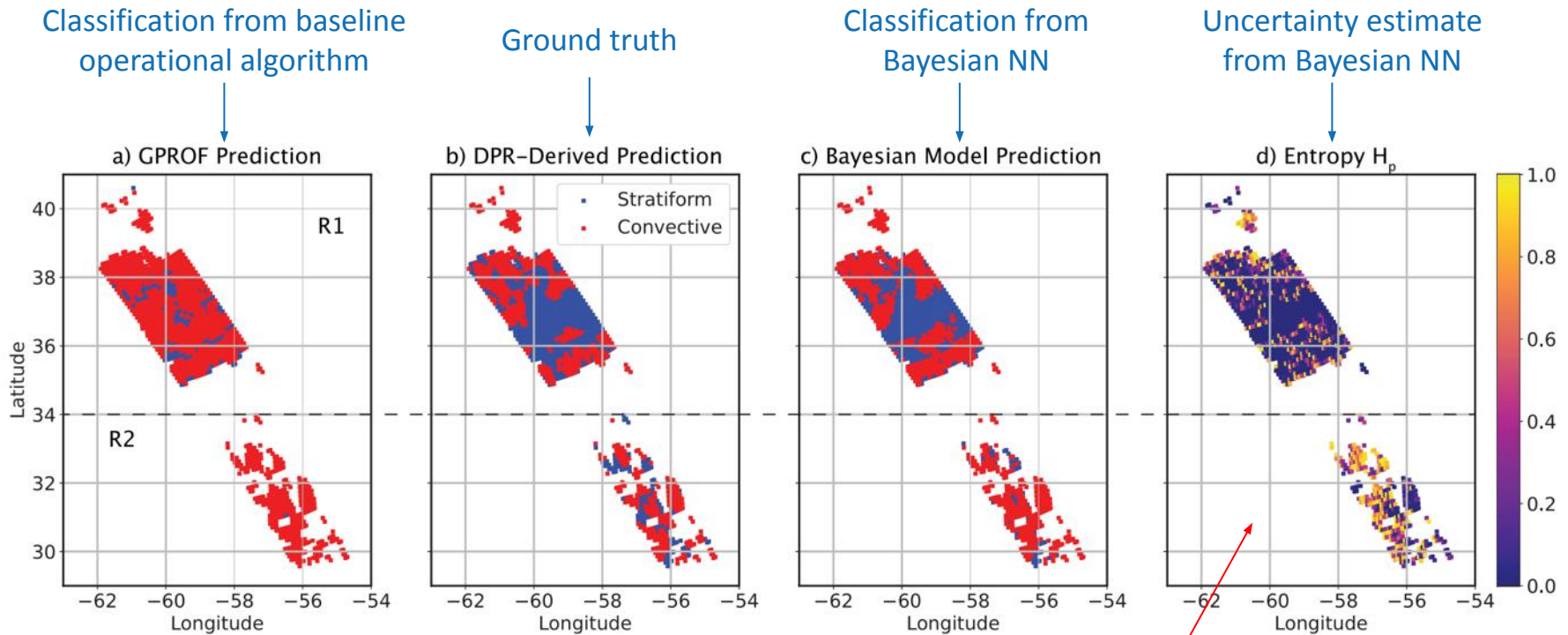


Fig. 1. Case study-August 11, 2018, GPM orbit 025293. From left to right: (a) GPROF prediction: a current operational benchmark. (b) DPR-derived product: the truth/label. (c) Bayesian model prediction ResNetV2. (d) Predictive H_p map (uncertainty map) for the Bayesian model prediction.

Comments on measures of uncertainty:

- Used here: entropy (high entropy = high uncertainty)
- Many alternatives exist, e.g. plot instead standard deviation.
- Orescanin et al. paper in progress on differences of using different measures.

Group 4: Depth of ocean mixed layer via probabilistic ML

Foster, D., D.J. Gagne II, and D.B. Whitt, 2021: “Probabilistic machine learning estimation of ocean mixed layer depth from dense satellite and sparse in-situ observations.” *Journal of Advances in Modeling Earth Systems*, **under review**.

You can find similar manuscript [here](#) or ask the authors for an up-to-date version.

Methods: They used 10 different ML methods (their Table 1), with 4 different methods for estimating uncertainty.

The 4 methods are:

- Direct sampling via dropout regularization
- Variational networks: assume that output follows parametric distribution, like Gaussian, then predict parameters of prediction
- Variational auto-encoders (VAE):
 - Couple VAE with a fully connected neural net, called the “estimator,” which transforms the latent space into an estimate of mixed-layer depth (MLD).
 - Because they use a Gaussian distribution in the latent space, the resulting MLD estimate is probabilistic.
- Deep ensemble: create ensemble of neural nets by training with different initial weights.

Application: Estimate depth of ocean mixed layer, based on satellite observations of sea-surface salinity, temperature, and height.

Group 4: Depth of ocean mixed layer via probabilistic ML

Comments:

- They used TensorFlow and TensorFlow probability for all methods.
- They used a wide variety of methods (4 methods for estimating uncertainty, 10 ML models overall).
- They evaluated deterministic predictions via the relative mean absolute error (MAE) and correlation.
- They evaluated uncertainty via the CRPS, Kullback-Leibler divergence, and Kolmogorov-Smirnov statistic -- all of which compare two probability distributions.

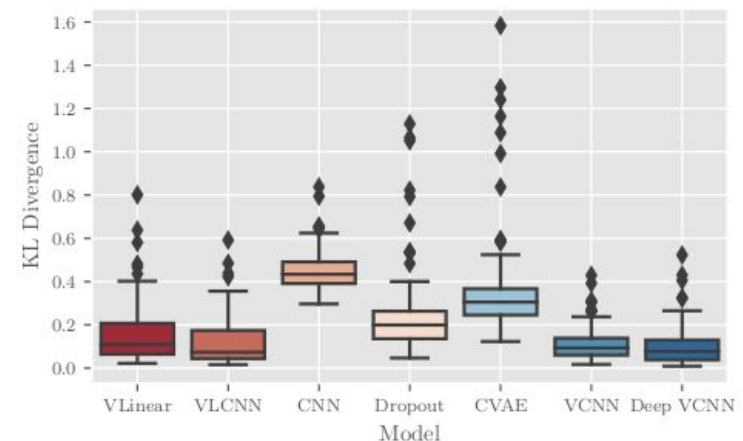
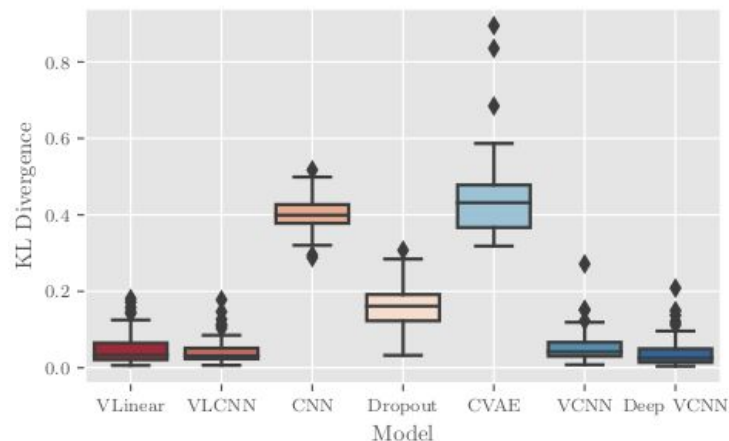
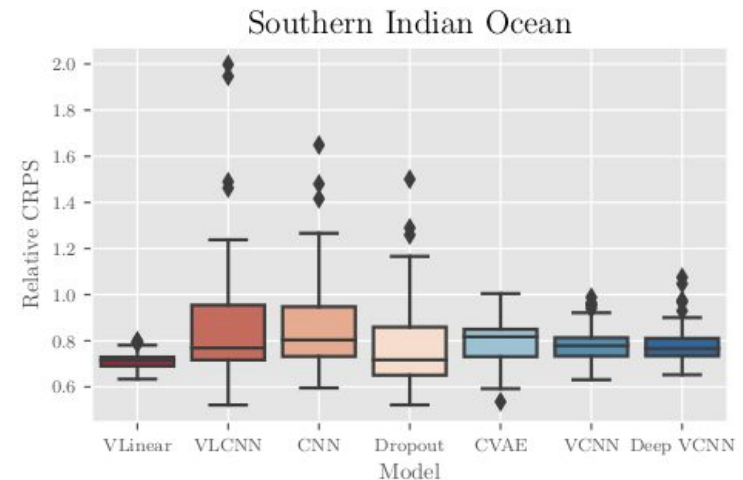
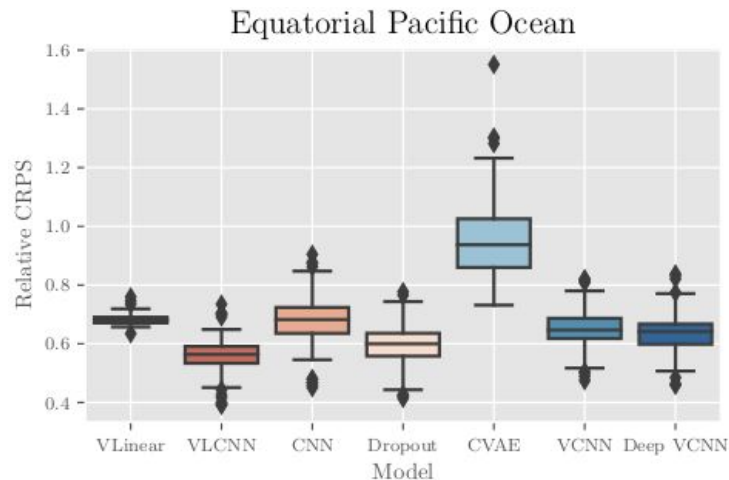
Key results:

- “[T]he machine learning algorithms produce reasonably well calibrated MLD [mixed-layer depth] estimates.”
- “[P]arameterized distributions outperform sampling- and ensemble-based uncertainty quantification techniques. This suggests that Gaussian parameterizations of the conditional uncertainty in MLD spatio-temporal variability is sufficient, but sampling techniques might also be improved in the future with additional data.”

Group 4: Depth of ocean mixed layer via probabilistic ML

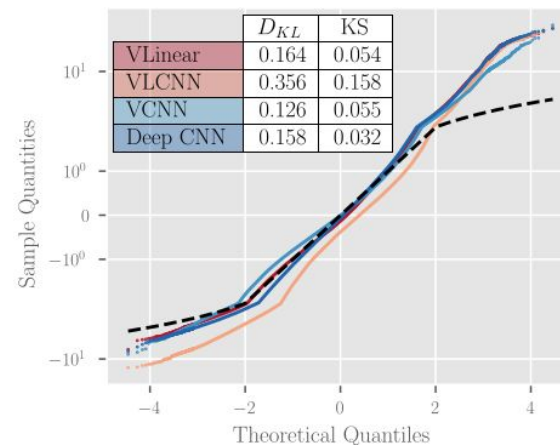
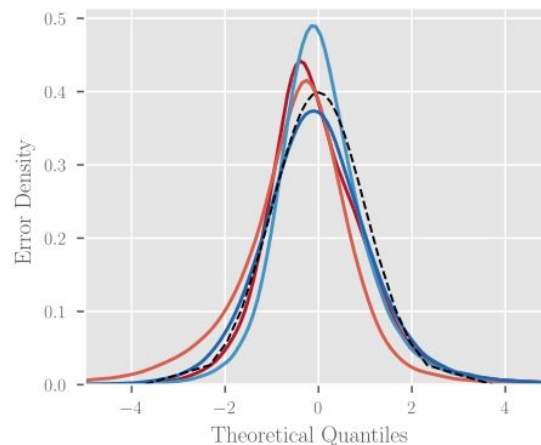
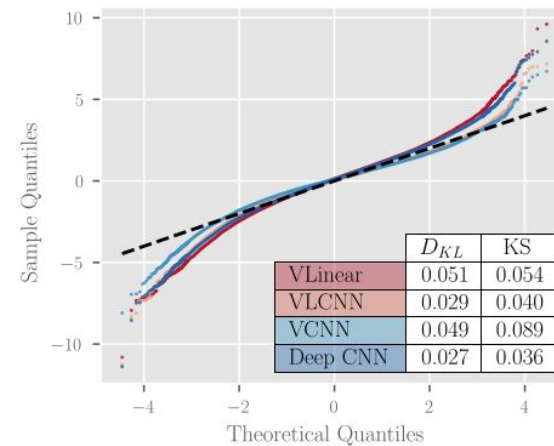
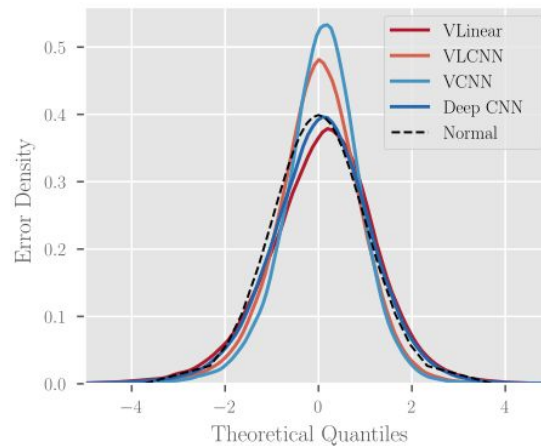
- **Results:**

- CRPS: VLCNN best in Pacific, VLinear in Indian Ocean
- KL divergence: variational methods (VLinear, VLCNN, VCNN, deep VCNN) best



Group 4: Depth of ocean mixed layer via probabilistic ML

- **Results:**
 - Pacific at top, Indian Ocean at bottom
 - A lot of mixed results here (see manuscript)
 - Figure shown mostly to demo another way uncertainty can be evaluated



Group 5: Bayesian DL for discrete-continuous and skewed data

Vandal, T., et al., 2018: “Quantifying uncertainty in discrete-continuous and skewed data with Bayesian deep learning.” *Proceedings of the 24th International Conference on Knowledge Discovery and Data-mining*, 2377-2386.

Methods:

- They use Bayesian deep learning (BDL) with two different parameterizations for the likelihoods: Gaussian and log-normal.
- They use BDL to estimate uncertainty for DeepSD, a CNN for super-resolution of precip data.
- Their approach accounts for both aleatory and epistemic uncertainty.
- They use variational inference (VI) for training, instead of Markov-chain Monte Carlo (MCMC).
- However, at inference time they do 50 passes of Monte Carlo sampling to estimate distribution.

Key results:

- BDL approach outperforms simple Gaussian parameterization, in terms of both deterministic prediction and uncertainty estimation.
- Log-normal distribution produces best uncertainty estimation at the extremes (can handle skewed data better).

Group 5: Bayesian DL for discrete-continuous and skewed data

- **Results:**
 - RMSE for deterministic predictions
 - “Gaussian” is based Gaussian parameterization, while “DC” is discrete-continuous, the new models suggested by the authors

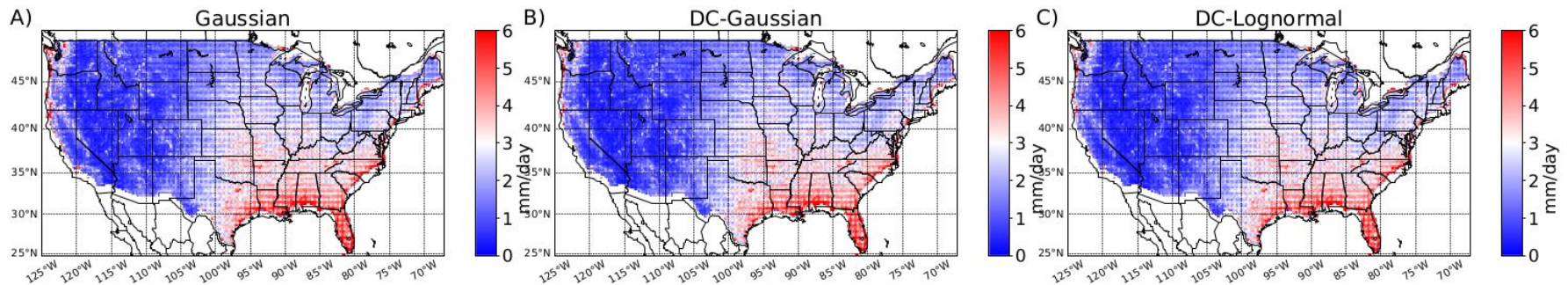


Figure 5: Daily Root Mean Square Error (RMSE) computed at each location for years 2006 to 2015 (test set) in CONUS. A) Gaussian, B) Conditional-Gaussian, and C) Conditional-Lognormal. Red corresponds to high RMSE while blue corresponds to low RMSE.

Group 5: Bayesian DL for discrete-continuous and skewed data

- **Results:**
 - Uncertainty estimation
 - DC-Lognormal performs best at the extremes (very few points outside the uncertainty range)

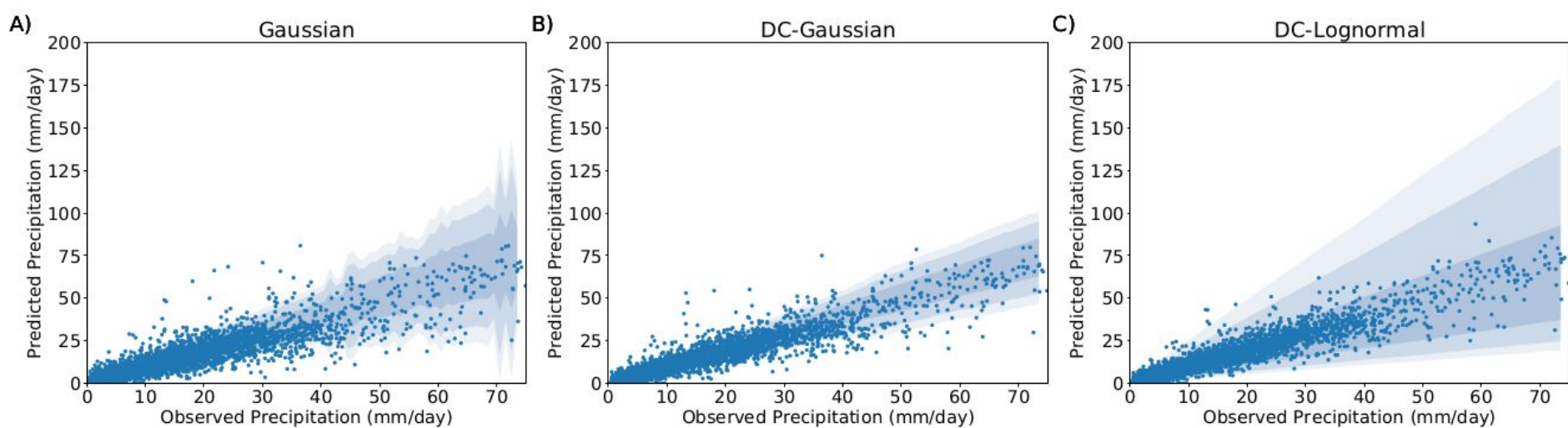


Figure 8: Uncertainty widths based on quantiles from their predictive distributions. The points are observations versus the expected value. The bands correspond to 50%, 80%, and 90% predictive intervals.

Other papers - not discussed here

Klotz, D., Kratzert, F., Gauch, M., Keefe Sampson, A., Brandstetter, J., Klambauer, G., Hochreiter, S. and Nearing, G., 2021. Uncertainty Estimation with Deep Learning for Rainfall–Runoff Modelling. Hydrology and Earth System Sciences Discussions, pp.1-32.

<https://hess.copernicus.org/preprints/hess-2021-154/hess-2021-154.pdf>

WeatherBench Probability: Medium-range weather forecasts with probabilistic machine learning methods

<https://ui.adsabs.harvard.edu/abs/2021EGUGA..2311448G/abstract>

Clare, M.C., Jamil, O. and Morcrette, C., 2021. Combining distribution-based neural networks to predict weather forecast probabilities. Quarterly Journal of the Royal Meteorological Society.

<https://rmets.onlinelibrary.wiley.com/doi/full/10.1002/qj.4180>