
# Neural Networks for Postprocessing Ensemble Weather Forecasts

STEPHAN RASP

*Meteorological Institute, Ludwig-Maximilians-Universität, Munich, Germany*

SEBASTIAN LERCH

*Institute for Stochastics, Karlsruhe Institute of Technology, Heidelberg Institute for Theoretical Studies, Karlsruhe, Germany*

(Manuscript received 23 May 2018, in final form 14 August 2018)

ABSTRACT

Ensemble weather predictions require statistical postprocessing of systematic errors to obtain reliable and accurate probabilistic forecasts. Traditionally, this is accomplished with distributional regression models in which the parameters of a predictive distribution are estimated from a training period. We propose a flexible alternative based on neural networks that can incorporate nonlinear relationships between arbitrary predictor variables and forecast distribution parameters that are automatically learned in a data-driven way rather than requiring prespecified link functions. In a case study of 2-m temperature forecasts at surface stations in Germany, the neural network approach significantly outperforms benchmark postprocessing methods while being computationally more affordable. Key components to this improvement are the use of auxiliary predictor variables and station-specific information with the help of embeddings. Furthermore, the trained neural network can be used to gain insight into the importance of meteorological variables, thereby challenging the notion of neural networks as uninterpretable black boxes. Our approach can easily be extended to other statistical postprocessing and forecasting problems. We anticipate that recent advances in deep learning combined with the ever-increasing amounts of model and observation data will transform the postprocessing of numerical weather forecasts in the coming decade.

## 1. Introduction

Numerical weather prediction based on physical models of the atmosphere has improved continuously since its inception more than four decades ago (Bauer et al. 2015). In particular, the emergence of ensemble forecasts—simulations with varying initial conditions and/or model physics—added another dimension by quantifying the flow-dependent uncertainty. Yet despite these advances the raw forecasts continue to exhibit systematic errors that need to be corrected using statistical postprocessing methods (Hemri et al. 2014). Considering the ever-increasing social and economical value of numerical weather prediction—for example, in the renewable energy industry—producing accurate and calibrated probabilistic forecasts is an urgent challenge.

Most postprocessing methods correct systematic errors in the raw ensemble forecast by learning a function that relates the response variable of interest to predictors. From a machine learning perspective, postprocessing can be viewed as a supervised learning task. For the purpose of this study we will consider postprocessing in a narrower distributional regression framework where the aim is to model the conditional distribution of the weather variable of interest given a set of predictors. The two most prominent approaches for probabilistic forecasts, Bayesian model averaging (BMA; Raftery et al. 2005) and nonhomogeneous regression, also referred to as ensemble model output statistics (EMOS; Gneiting et al. 2005), rely on parametric forecast distributions. This means one has to specify a predictive distribution and estimate its parameters, for example, the mean and the standard deviation in the case of a Gaussian distribution. Within the

Denotes content that is immediately available upon publication as open access.

Supplemental information related to this paper is available at the Journals Online website: https://doi.org/10.1175/MWR-D-18-0187.s1.

*Corresponding author*: Stephan Rasp, s.rasp@lmu.de

DOI: 10.1175/MWR-D-18-0187.1

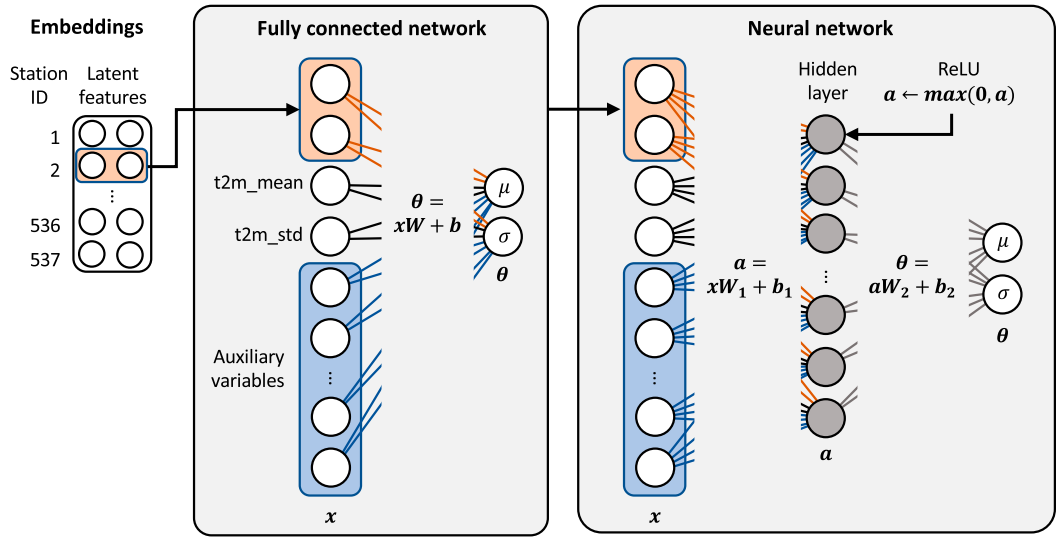


FIG. 1. Schematic of (left) an FCN and (right) an NN with one hidden layer. In both cases, data flow from left to right. Orange nodes and connections illustrate station embeddings, and blue nodes are for auxiliary input variables. Mathematical operations are to be understood as elementwise operations for vector objects.

EMOS framework the distribution parameters are connected to summary statistics of the ensemble predictions through suitable link functions that are estimated by minimizing a probabilistic loss function over a training dataset. Including additional predictors, such as forecasts of cloud cover or humidity, is not straightforward within this framework and requires elaborate approaches to avoid overfitting (Messner et al. 2017), a term that describes the inability of a model to generalize to data outside the training dataset. We propose an alternative approach based on modern machine learning methods, which is capable of including arbitrary predictors and learns nonlinear dependencies in a data-driven way.

Much work over the past years has been spent on flexible machine learning techniques for statistical modeling and forecasting (McGovern et al. 2017). Random forests (Breiman 2001), for instance, can model nonlinear relationships including arbitrary predictors while being robust to overfitting. They have been used for the classification and prediction of precipitation (Gagne et al. 2014), severe wind (Lagerquist et al. 2017), and hail (Gagne et al. 2017). Within a postprocessing context, quantile regression forest models have been proposed by Taillardat et al. (2016).

Neural networks are a flexible and user-friendly machine learning algorithm that can model arbitrary nonlinear functions (Nielsen 2015). They consist of several layers of interconnected nodes that are modulated with simple nonlinearities (Fig. 1; section 4). Over the past decade many fields, most notably computer vision and natural language processing (LeCun et al. 2015), but also biology, physics, and chemistry (Angermueller et al. 2016; Goh et al. 2017), have been transformed by neural networks. In the atmospheric sciences, neural networks have been used to detect extreme weather in climate datasets (Liu et al. 2016) and parameterize subgrid processes in general circulation models (Gentine et al. 2018; Rasp et al. 2018). Neural networks have also been used for forecasting solar irradiances (Wang et al. 2012; Chu et al. 2013) and damaging winds (Lagerquist et al. 2017). However, the complexity of the neural networks used in these studies was limited.

Here, we demonstrate how neural networks can be used for probabilistic postprocessing of ensemble forecasts within the distributional regression framework. The presented model architecture allows for the incorporation of various features that are relevant for correcting systematic deficiencies of ensemble predictions, and to estimate the network parameters by optimizing the continuous ranked probability score—a mathematically principled loss function for probabilistic forecasts. Specifically, we explore a case study of 2-m temperature forecasts at surface stations in Germany with data from 2007 to 2016. We compare different neural network configurations to benchmark postprocessing methods for varying training period lengths. We further use the trained neural networks to gain meteorological insight into the problem at hand. Our ultimate goal is to present an efficient, multipurpose approach to statistical postprocessing and probabilistic forecasting. To the best of our knowledge, this study is the first to tackle ensemble postprocessing using neural networks.

The remainder of the paper is structured as follows. Section 2 describes the forecast and observation data as well as the notation used throughout the study. In section 3 we describe the benchmark postprocessing models, followed by a description of the neural network techniques in section 4. The main results are presented in section 5. In section 6 we explore the relative importance of the predictor variables. A discussion of possible extensions follows in section 7 before our conclusions are presented in section 8.

Python (Python Software Foundation 2017) and R (R Core Team 2017) code for reproducing the results is available online (https://github.com/slerch/ppnn).

## 2. Data and notation

### a. Forecast data

For this study, we focus on 2-m temperature forecasts at surface stations in Germany at a forecast lead time of 48 h. The forecasts are taken from the THORPEX Interactive Grand Global Ensemble (TIGGE) dataset[1] (Bougeault et al. 2010). In particular, we use the global European Centre for Medium-Range Weather Forecasts (ECMWF) 50-member ensemble forecasts initialized at 0000 UTC every day. The data in the TIGGE archive are upscaled onto a $0.5° \times 0.5°$ grid, which corresponds to a horizontal grid spacing of around 35/55 km (zonal/meridional). For comparison with the station observations, the gridded data were bilinearly interpolated to the observation locations. In addition to the target variable, we retrieved several auxiliary predictor variables (Table 1[2]). These were chosen broadly based on meteorological intuition.[3] For each variable, we reduced the 50-member ensemble to its mean and standard deviation.

Ensemble predictions are available from 3 January 2007 to 31 December 2016 every day. For model estimation we use two training periods, 2007–15 and 2015 only, to assess the importance of training sample size. To validate the performance of the different models correctly, it is important to mimic operational conditions as closely as possible. For this reason we chose future dates only, in our case the entire year 2016, rather than a random subsample of the entire dataset. Note also that the ECMWF forecasting system has undergone major changes during this 10-yr period. This might counteract the usefulness of using longer training periods.

TABLE 1. Abbreviations and descriptions of all features.

| Feature | Description |
|---|---|
| Ensemble predictions (mean and std dev) | |
| t2m | 2-m temperature |
| cape | Convective available potential energy |
| sp | Surface pressure |
| tcc | Total cloud cover |
| sshf | Sensible heat flux |
| slhf | Latent heat flux |
| u10 | 10-m *U* wind |
| v10 | 10-m *V* wind |
| d2m | 2-m dewpoint temperature |
| ssr | Shortwave radiation flux |
| str | Longwave radiation flux |
| sm | Soil moisture |
| u_pl500 | *U* wind at 500 hPa |
| v_pl500 | *V* wind at 500 hPa |
| u_pl850 | *U* wind at 850 hPa |
| v_pl850 | *V* wind at 850 hPa |
| gh_pl500 | Geopotential at 500 hPa |
| q_pl850 | Specific humidity at 850 hPa |
| Station-specific information | |
| station_alt | Altitude of station |
| orog | Altitude of model grid point |
| station_lat | Lat of station |
| station_lon | Lon of station |

### b. Observation data

The forecasts are evaluated at 537 weather stations in Germany (see Fig. 2[4]). The 2-m temperature data are available from the Climate Data Center of the German Weather Service [Deutscher Wetterdienst (DWD)[5]]. Several stations have periods of missing data, which are omitted from the analysis. During the evaluation period in calendar year 2016, observations are available at 499 stations.

After removing missing observations, the 2016 validation set contains 182 218 samples, the 2007–15 training set contains 1 626 724 samples, and the 2015 training set contains 180 849 samples.

### c. Notation

We now introduce the notation that is used throughout the rest of the paper. An observation of 2-m temperature

[1] Available at http://apps.ecmwf.int/datasets/data/tigge/, see https://github.com/slerch/ppnn/tree/master/data_retrieval.

[2] Detailed definitions are available at https://software.ecmwf.int/wiki/display/TIGGE/Parameters.

[3] Similar sets of predictors have been used, for example, in Messner et al. (2017), Schlosser et al. (2018), and Taillardat et al. (2016, 2017).

[4] All maps in this article were produced using the R package ggmap (Kahle and Wickham 2013).

[5] Available at https://www.dwd.de/DE/klimaumwelt/cdc/cdc_node.html.

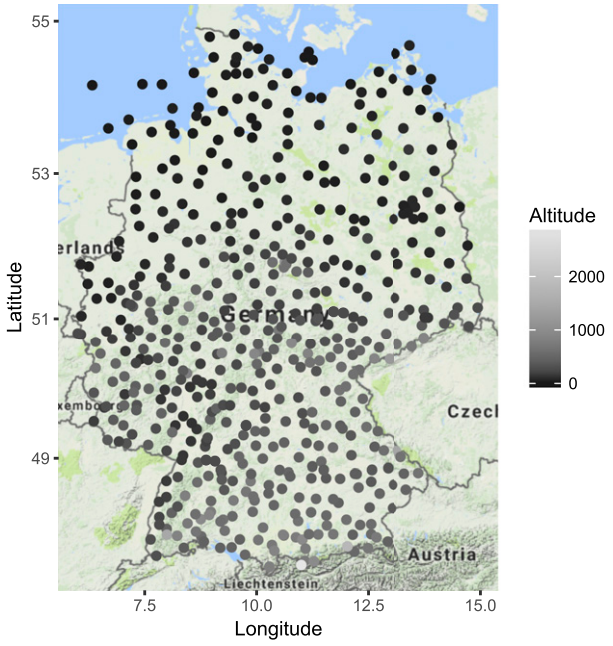


FIG. 2. Locations of DWD surface observation stations. The grayscale values of the points indicate the altitude (m).

at station $s \in \{1, \ldots, S\}$ and time $t \in \{1, \ldots, T\}$ will be denoted by $y_{s,t}$. For each $s$ and $t$, the 50-member ECMWF ensemble forecast of variable $v$ is given by $x_{s,t}^{v,1}, \ldots, x_{s,t}^{v,50}$, with mean value $x_{s,t}^{v,\text{mean}}$ and standard deviation $x_{s,t}^{v,\text{sd}}$. The mean values and standard deviations of all variables in the top part of Table 1 are combined with station-specific features in the bottom part, and aggregated into a vector of predictors $\mathbf{X}_{s,t} \in \mathbb{R}^p$, $p = 42$. Further, we write $\mathbf{X}_{s,t}^{\text{t2m}}$ to denote the vector of predictors that only contains the mean value and standard deviation of the 2-m temperature forecasts.

## 3. Benchmark postprocessing techniques

### a. *Ensemble model output statistics*

Within the general EMOS framework proposed by Gneiting et al. (2005), the conditional distribution of the weather variable of interest, $y_{s,t}$, given ensemble predictions $\mathbf{X}_{s,t}$, is modeled by a single parametric forecast distribution $F_{\boldsymbol{\theta}_{s,t}}$ with parameters $\boldsymbol{\theta}_{s,t} \in \mathbb{R}^d$:

$$y_{s,t} | \mathbf{X}_{s,t} \sim F_{\boldsymbol{\theta}_{s,t}}. \tag{1}$$

The parameters vary over space and time, and depend on the ensemble predictions $\mathbf{X}_{s,t}$ through suitable link functions $g : \mathbb{R}^p \to \mathbb{R}^d$:

$$\boldsymbol{\theta}_{s,t} = g(\mathbf{X}_{s,t}). \tag{2}$$

Here, we are interested in modeling the conditional distribution of temperature and follow Gneiting et al. (2005), who introduced a model based on ensemble predictions of temperature, $\mathbf{X}_{s,t}^{\text{t2m}}$, only, where the forecast distribution is Gaussian with parameters $\boldsymbol{\theta}_{s,t} \in \mathbb{R}^2$ given by mean $\mu_{s,t}$ and standard deviation $\sigma_{s,t}$, that is,

$$y_{s,t} \Big| \mathbf{X}_{s,t}^{\text{t2m}} \sim \mathcal{N}_{(\mu_{s,t}, \sigma_{s,t})},$$

and where the link functions for the mean and standard deviation are affine functions of the ensemble mean and standard deviation, respectively:

$$(\mu_{s,t}, \sigma_{s,t}) = g(\mathbf{X}_{s,t}^{\text{t2m}}) = (a_{s,t} + b_{s,t} x_{s,t}^{\text{t2m,mean}}, c_{s,t} + d_{s,t} x_{s,t}^{\text{t2m,sd}}). \tag{3}$$

Over the past decade, the EMOS framework has been extended from temperature to other weather variables including wind speed (Thorarinsdottir and Gneiting 2010; Lerch and Thorarinsdottir 2013; Baran and Lerch 2015; Scheuerer and Möller 2015) and precipitation (Messner et al. 2014; Scheuerer 2014; Scheuerer and Hamill 2015).

The model parameters (or EMOS coefficients) $\boldsymbol{\kappa}_{s,t} = (a_{s,t}, b_{s,t}, c_{s,t}, d_{s,t})$ are estimated by minimizing the mean continuous ranked probability score (CRPS) as a function of the parameters over a training set. The CRPS is an example of a proper scoring rule (i.e., a mathematically principled loss function for distribution forecasts) and is a standard choice in meteorological applications. Details on the mathematical background of proper scoring rules and their use for model estimation are provided in the appendix.

Training sets are often considered to be composed of the most recent days only. However, as we did not find substantial differences in predictive performance, we estimate the coefficients over a fixed training set, they thus do not vary over time and we denote them by $\boldsymbol{\kappa}_s$. Estimation is usually either performed locally (i.e., considering only forecast cases from the station of interest) or globally by pooling together forecasts and observations from all stations. We refer to the corresponding EMOS models as EMOS-loc and EMOS-gl, respectively. The parameters $\boldsymbol{\kappa}$ of the global model do not depend on the station $s$ and are, thus, unable to correct location-specific deficiencies of the ensemble forecasts. Alternative approaches where training sets are selected based on similarities of weather situations or observation station characteristics were proposed by Junk et al. (2015) and Lerch and Baran (2017). Both EMOS-gl and EMOS-loc are implemented in R with the help of the scoringRules package (Jordan et al. 2018).

### b. Boosting for predictor selection in EMOS models

Extending the EMOS framework to allow for including additional predictor variables is nontrivial as the increased number of parameters can result in overfitting. Messner et al. (2017) proposed a boosting algorithm for this purpose. In this approach components of the link function $g$ in (2) are chosen to be an affine function for the mean $\mu_{s,t}$ and an exponential transformation of an affine function for the standard deviation $\sigma_{s,t}$:

$$(\mu_{s,t}, \sigma_{s,t}) = g(\mathbf{X}_{s,t}) = \{(1, \mathbf{X}_{s,t})^{\mathrm{T}} \boldsymbol{\beta}_{s,t}, \exp[(1, \mathbf{X}_{s,t})^{\mathrm{T}} \boldsymbol{\gamma}_{s,t}]\}. \tag{4}$$

Here, $\boldsymbol{\beta}_{s,t} \in \mathbb{R}^{p+1}$ and $\boldsymbol{\gamma}_{s,t} \in \mathbb{R}^{p+1}$ denote coefficient vectors corresponding to the vector of predictors $\mathbf{X}_{s,t}$ extended by a constant. As for the standard EMOS models, the coefficient vectors are estimated over fixed training periods and thus do not depend on $t$; we suppress the index in the following.

The boosting algorithm proceeds iteratively by updating the coefficient of the predictor that improves the current model fit most. As the coefficient vectors are initialized as $\boldsymbol{\beta}_s = \boldsymbol{\gamma}_s = 0$, only the most important variables will have nonzero coefficients if the algorithm is stopped before convergence. The contributions of the different predictors are assessed by computing average correlations to partial derivatives of the loss function with respect to $\mu_{s,t}$ and $\sigma_{s,t}$ over the training set. If the current model fit is improved, the coefficient vectors are updated by a predefined step size into the direction of steepest descent of linear approximations of the gradients.

We denote local EMOS models with an additional boosting step by EMOS-loc-bst. The tuning parameters of the algorithm were chosen by fitting models for a variety of choices and picking the configuration with the best out-of-sample predictions (see the online supplemental material) based on implementations in the R package crch (Messner et al. 2016). Note, however, that the results are not very sensitive to the exact choice of tuning parameters. For the local model considered here, the station-specific features in the bottom part of Table 1 are not relevant and are excluded from $\mathbf{X}_{s,t}$. Boosting-based variants of global EMOS models have also been tested, but result in worse forecasts.

The boosting-based EMOS-loc-bst model differs from the standard EMOS models (EMOS-gl and EMOS-loc) in several aspects. First, the boosting step allows us to include covariate information from predictor variables other than temperature forecasts. Second, the parameters are estimated by maximum likelihood estimation (i.e., by minimizing the mean logarithmic score by contrast to minimum CRPS estimation; see the appendix for details).[6] Further, the affine link function for the standard deviation in (3) is replaced by an affine function for the logarithm of the standard deviation in (4). By construction the boosting-based EMOS approach is unable to model interactions of the predictors. In principle, including nonlinear combinations (e.g., products) of predictors as additional input allows us to introduce such effects; however, initial tests indicated no substantial improvements.

### c. Quantile regression forests

Parametric distributional regression models such as the EMOS methods described above require the choice of a suitable parametric family $F_{\boldsymbol{\theta}}$. While the conditional distribution of temperature can be well approximated by a Gaussian distribution, this poses a limitation for other weather variables such as wind speed or precipitation where the choice is less obvious (see, e.g., Baran and Lerch 2018).

Nonparametric distributional regression approaches provide alternatives that circumvent the choice of the parametric family. For example, quantile regression approaches approximate the conditional distribution by a set of quantiles. Within the context of postprocessing ensemble forecasts, Taillardat et al. (2016) proposed a quantile regression forest (QRF) model based on the work of Meinshausen (2006) that allows us to include additional predictor variables.

The QRF model is based on the idea of generating random forests from classification and regression trees (Breiman et al. 1984). These are binary decision trees obtained by iteratively splitting the training data into two groups according to some threshold for one of the predictors, chosen such that every split minimizes the sum of the variance of the response variable in each of the resulting groups. The splitting procedure is iterated until a stopping criterion is reached. The final groups (or terminal leaves) thus contain subsets of the training observations based on the predictor values, and out-of-sample forecasts at station $s$ and time $t$ can be obtained by proceeding through the decision tree according to the corresponding predictor values $\mathbf{X}_{s,t}$. Random forest models (Breiman 2001) increase the stability of the predictions by averaging over many random decision trees generated by selecting a random subset of the

[6] A recent development version of the R package crch provides implementations of CRPS-based model estimation and boosting. However, initial tests indicated slightly worse predictive performance; we thus focus on maximum likelihood-based methods instead.

predictors at each candidate split in conjunction with bagging (i.e., bootstrap aggregation of random subsamples of training sets). In the quantile regression forest approach, each tree provides an approximation of the distribution of the variable of interest given by the empirical cumulative distribution function (CDF) of the observation values in the terminal leaf associated with the current predictor values $\mathbf{X}_{s,t}$. Quantile forecasts can then be computed from the combined forecast distribution, which is obtained by averaging over all tree-based empirical CDFs.

We implement a local version of the QRF model where separate models are estimated for each station based on training sets that only contain past forecasts and observations from that specific station. As discussed by Taillardat et al. (2016), the predicted quantiles are necessarily restricted to the range of observed values in the training period by construction, which may be disadvantageous in cases of shorter training periods. However, global variants of the QRF model did not result in improved forecast performance even with only one year of training data; we will thus restrict attention to the local QRF model. The models are implemented using the quantregForest package (Meinshausen 2017) for R. Tuning parameters are chosen as for the EMOS-loc-bst model (see the supplemental material).

The QRF approach has recently been extended in several directions. Athey et al. (2016) propose a generalized version of random forest-based quantile regression based on theoretical considerations (GRF), which has been tested but did not result in improved forecast performance. Taillardat et al. (2017) combine QRF (and GRF) models and parametric distributional regression by fitting a parametric CDF to the observations in the terminal leaves instead of using the empirical CDF. Schlosser et al. (2018) combine parametric distributional regression and random forests for parameter estimation within the framework of a generalized additive model for location, scale, and shape.

## 4. Neural networks

In this section we will give a brief introduction to neural networks. For a more detailed treatment the interested reader is referred to more comprehensive resources (e.g., Nielsen 2015; Goodfellow et al. 2016). The network techniques are implemented using the Python libraries Keras (Chollet et al. 2015) and TensorFlow (Abadi et al. 2016).

Neural networks consist of several layers of nodes (Fig. 1), each of which is a weighted sum of all nodes $j$ from the previous layer plus a bias term:

$$\sum_j w_j x_j + b \,. \tag{5}$$

The first layer contains the input values, or features, while the last layer represents the output values, or targets. In the layers in between, called hidden layers, each node value is passed through a nonlinear activation function. For this study, we use a rectified linear unit (ReLU):

$$\mathrm{ReLU}(x) = \max(0, x) \,.$$

This activation function allows the neural network to represent nonlinear functions. We tried other common nonlinear activation functions, such as sigmoid or hyperbolic tangent, but obtained the best results with ReLUs, which are the first choice for most applications these days. The weights and biases are optimized to reduce a loss function using stochastic gradient descent (SGD). Here, we employ an SGD version called Adam (Kingma and Ba 2014).

In this study we use networks without a hidden layer and with a single hidden layer (Fig. 1). The former, which we will call fully connected networks (FCNs), model the outputs as a linear combination of the inputs. The latter, called neural networks (NNs) here, are capable of representing nonlinear relationships and interactions. Introducing additional hidden layers to neural networks did not improve the predictions as additional model complexity increases the potential of overfitting. For more details on network hyperparameters, see the supplemental material.

### a. *Neural networks for ensemble postprocessing*

Neural networks can be applied to a range of problems, such as regression and classification. The main difference between those options is in the contents and activation function of the output layer, as well as the loss function. Here, we use the neural network for the distributional regression task of postprocessing ensemble forecasts. Our output layer represents the distribution parameters $\mu_{s,t}$ and $\sigma_{s,t}$ of the Gaussian predictive distribution. No activation function is applied. The corresponding probabilistic forecast describes the conditional distribution of the observation $y_{s,t}$ given the predictors $\mathbf{X}_{s,t}$ as input features. As a loss function for determining the network parameters, we use the closed form expression of the CRPS for a Gaussian distribution; see (A2). This is a nonstandard choice in the neural network literature [D'Isanto and Polsterer (2018) is the only previous study to our knowledge] but provides a mathematically principled choice for the distributional regression problem at hand (see the appendix for the mathematical background). Other probabilistic neural

network approaches include quantile regression (Taylor 2000) and distribution-to-distribution regression (Kou et al. 2018).

The simplest network model is a fully connected model based on predictors $\mathbf{X}_{s,t}^{\text{t2m}}$ [i.e., mean and standard deviation of ensemble predictions of temperature only (denoted by FCN)]. Apart from additional connections for the mean and standard deviation to the ensemble standard deviation and mean, respectively, the FCN model is conceptually equivalent to EMOS-gl, but differs in the parameter estimation approaches. A neural network with a hidden layer for the $\mathbf{X}_{s,t}^{\text{t2m}}$ input did not show any improvements over the simple linear model, suggesting that there are no nonlinear relationships to exploit. Additional information from auxiliary variables can be taken into account by considering the entire vector $\mathbf{X}_{s,t}$ of predictors as input features. The corresponding fully connected and neural network models are referred to as FCN-aux and NN-aux.

### b. Station embeddings

To enable the networks to learn station-specific information, we use embeddings, a common technique in natural language processing and recommender systems. An embedding $e$ is a mapping from a discrete object, in our case the station ID $s$, to a vector of real numbers $\mathbf{X}_s^{\text{emb}}$ (Guo and Berkhahn 2016):

$$e : s \mapsto \mathbf{X}_s^{\text{emb}} ,$$

where $\mathbf{X}_s^{\text{emb}} \in \mathbb{R}^{n_{\text{emb}}}$; $n_{\text{emb}}$ is the number of elements in the embedding vector which are also referred to as latent features. These latent features encode information about each station $s$ but do not correspond to any real variable. In total then, the embedding matrix has dimension $S \times n_{\text{emb}}$, where $S$ is the number of stations. The latent features $\mathbf{X}_s^{\text{emb}}$ are concatenated with the predictors, $\mathbf{X}_{s,t}^{\text{t2m}}$ or $\mathbf{X}_{s,t}$, and are updated along with the weights and biases during training. This allows the algorithm to learn a specific set of numbers for each station. Here, we use $n_{\text{emb}} = 2$ because larger values did not improve the predictions.

The fully connected network with input features $\mathbf{X}_{s,t}^{\text{t2m}}$ and embeddings is abbreviated by FCN-emb. As with FCN, adding a hidden layer did not improve the results. Fully connected and neural networks with both, station embeddings and auxiliary inputs $\mathbf{X}_{s,t}$, are denoted by FCN-aux-emb and NN-aux-emb.

### c. Further network details

Neural networks with a large number of parameters (i.e., weights and biases) can suffer from overfitting. One way to reduce overfitting is to stop training early. When to stop can be guessed by taking out a subset (20%) from the training set (2007–15 or 2015) and checking when the score on this separate dataset stops improving. This gives a good approximation of when to stop training on the full training set without using the actual 2016 validation set during training. Other common regularization techniques to prevent overfitting, such as dropout or weight decay (L2 regularization), were not successful in our case for reasons unclear to us. Further investigation in follow-on studies may be helpful.

Finally, we train ensembles of 10 neural networks with different random initial parameters for each configuration and average over the forecast distribution parameter estimates to obtain $\boldsymbol{\theta}_{s,t}$. For the more complex network models this helps to stabilize the parameter estimates by reducing the variability due to random variations between model runs and slightly improves the forecasts.

## 5. Results

Tuning parameters for all benchmark and network models are listed in the supplemental material (Tables S1 and S2). Details on the employed evaluation methods are provided in the appendix.

### a. General results

The CRPS values averaged over all stations and the entire 2016 validation period are summarized in Table 2.[7] For the 2015 training period, EMOS-gl gives a 13% relative improvement compared to the raw ECMWF ensemble forecasts in terms of mean CRPS. As expected, FCN, which mimics the design of EMOS-gl, achieves a very similar score. Adding local station information in EMOS-loc and FCN-emb improves the global score by another 10%. While EMOS-loc estimates a separate model for each station, FCN-emb can be seen as a global network–based implementation of EMOS-loc. Adding covariate information through auxiliary variables results in an improvement for the fully connected models similar to that of adding station information. Combining auxiliary variables and station embeddings in FCN-emb-aux improves the mean CRPS further to 0.88 but the effects do not stack linearly. Adding covariate information in EMOS models using boosting (EMOS-loc-bst) outperforms FCN-emb-aux by 3%. Allowing for nonlinear interactions of station information and auxiliary variables using a neural

[7] To account for the intertwined choice of scoring rules for model estimation and evaluation (Gebetsberger et al. 2017), we have also evaluated the models using LogS. However, as the results are very similar to those reported here and computation of LogS for the raw ensemble and QRF forecasts is problematic (Krüger et al. 2016), we focus on CRPS-based evaluation.

TABLE 2. Mean CRPSs for raw and postprocessed ECMWF ensemble forecasts, averaged over all available observations during calendar year 2016. The lowest (i.e., best) values are marked in boldface.

| Model | Description | Mean CRPS for training period | |
|---|---|---|---|
| | | 2015 | 2007–15 |
| Raw ensemble | | 1.16 | 1.16 |
| Benchmark postprocessing methods | | | |
| EMOS-gl | Global EMOS | 1.01 | 1.00 |
| EMOS-loc | Local EMOS | 0.90 | 0.90 |
| EMOS-loc-bst | Local EMOS with boosting | 0.85 | 0.80 |
| QRF | Local quantile regression forest | 0.95 | 0.81 |
| Neural network models | | | |
| FCN | Fully connected network | 1.01 | 1.01 |
| FCN-aux | . . .with auxiliary predictors | 0.92 | 0.91 |
| FCN-emb | . . .with station embeddings | 0.91 | 0.91 |
| FCN-aux-emb | . . .with both of the above | 0.88 | 0.87 |
| NN-aux | One-hidden-layer NN with auxiliary predictors | 0.90 | 0.86 |
| NN-aux-emb | . . .and station embeddings | **0.82** | **0.78** |

network (NN-aux-emb) achieves the best results, improving the best benchmark technique (EMOS-loc-bst) by 3% for a total improvement compared to the raw ensemble of 29%. The QRF model is unable to compete with most of the postprocessing models for the 2015 training period.

The relative scores and model rankings for the 2007–15 training period closely match those of the 2015 period. For the linear models (EMOS-gl, EMOS-loc, and all FCN) more data does not improve the score by much. For EMOS-loc-bst and the neural network models, however, the skill is increased by 4%–5%. This suggests that longer training periods are most efficiently exploited by more complex, nonlinear models. QRF improves the most, now being among the best models, which indicates a minimum data amount required for this method to work. This is likely due to the limitation of predicted quantiles to the range of observed values in the training data; see section 3c.

To assess calibration, verification rank and probability integral transform (PIT) histograms of raw and postprocessed forecasts are shown in the supplemental material. The raw ensemble forecasts are underdispersed, as indicated by the U-shaped verification rank histogram; that is, observations tend to fall outside the range of the ensemble too frequently. By contrast, all postprocessed forecast distributions are substantially better calibrated and the corresponding PIT histograms show much smaller deviations from uniformity. All models show a slight overprediction of high temperatures and, with the exception of QRF, an underprediction of low values. This might be due to residual skewness (Gebetsberger et al. 2018). The linear EMOS and FCN models as well as QRF are further slightly overdispersive, as indicated by the inverse U-shaped top parts of the histogram.

### b. *Station-by-station results*

Figure 3 shows the station-wise distribution of the continuous ranked probability skill score (CRPSS), which measures the probabilistic skill relative to a reference model. Positive values indicate an improvement over the reference. Compared to the raw ensemble, forecasts at most stations are improved by all postprocessing methods with only a few negative outliers. Compared to EMOS-loc, only FCN-aux-emb, the neural network models, and EMOS-loc-bst show improvements at the majority of the stations. Corresponding plots with the three best-performing models as reference experiments are provided in the supplemental material. It is interesting to note that the network models, with the exception of FCN and FCN-emb, have more outliers, particularly for negative values compared to the EMOS methods and QRF, which have very few negative outliers. This might be due to a few stations with strongly location-specific error characteristics that the locally estimated benchmark models are better able to capture. Training with data from 2007 to 2015 alleviates this somewhat.

Figure 4 shows maps with the best-performing models in terms of mean CRPS for each station. For the majority of stations NN-aux-emb provides the best predictions. The variability of station-specific best models is greater for the 2015 training period compared to 2007–15. The top three models for the 2015 period are NN-aux-emb (best at 65.9% of stations), EMOS-loc-bst (16.0%), and NN-aux (7.2%), and for 2007–15 they are

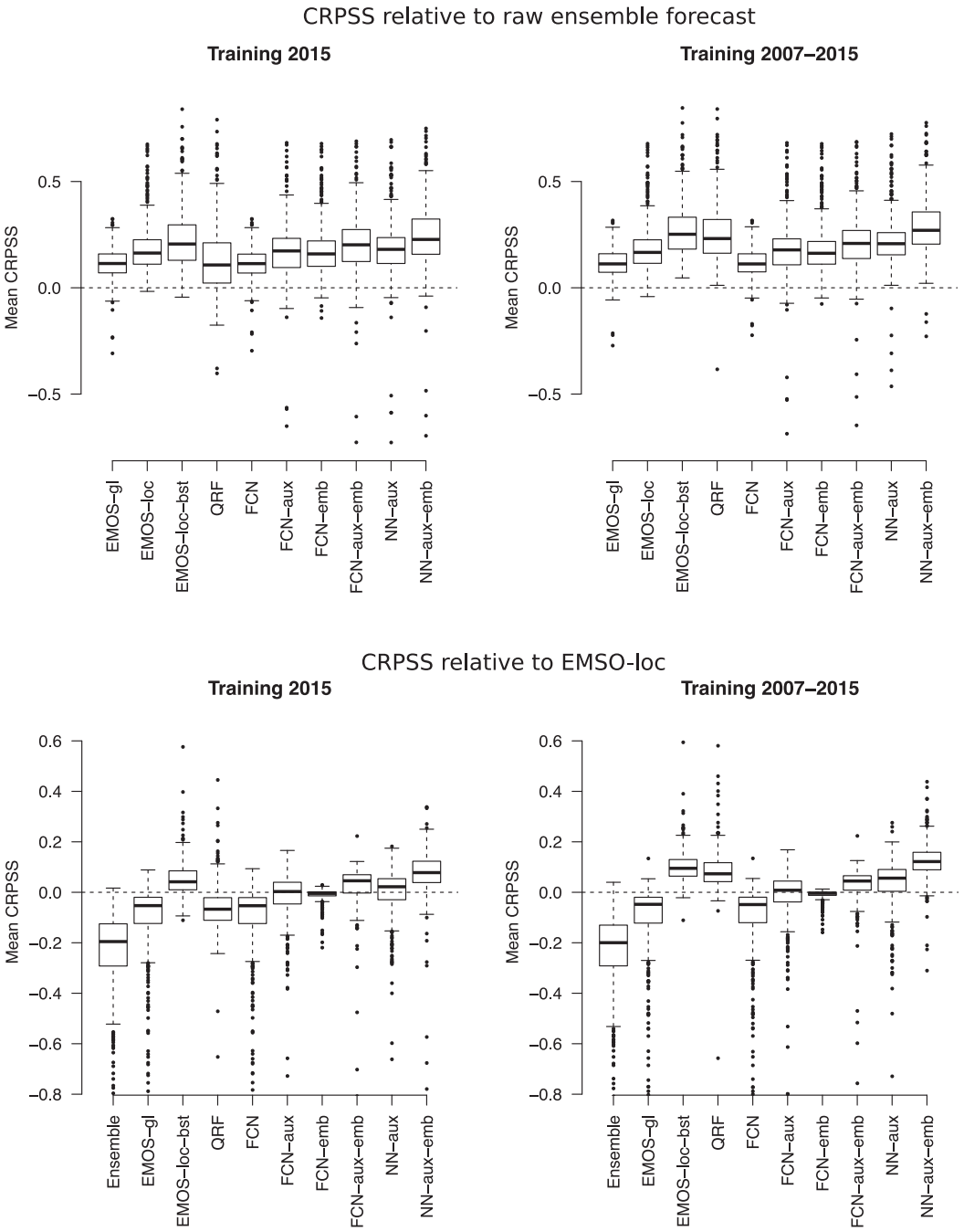


FIG. 3. Boxplots of stationwise mean CRPSS of all postprocessing models using the (top) raw ensemble and (bottom) EMOS-loc as the reference forecast. A dot within each box represents the mean CRPSS at one of the observation stations. The CRPSS is computed so that positive values indicate an improvement of the model specified on the horizontal axis over the reference. Similar plots with different reference models are provided in the supplemental material.

NN-aux-emb (73.5%), EMOS-loc-bst (12.4%), and QRF (7.4%). At coastal and offshore locations, particularly for the shorter training period, the benchmark methods tend to outperform the network methods. Ensemble forecast errors at these locations likely have a strong location-specific component that might be easier to capture for the locally estimated EMOS and QRF methods.

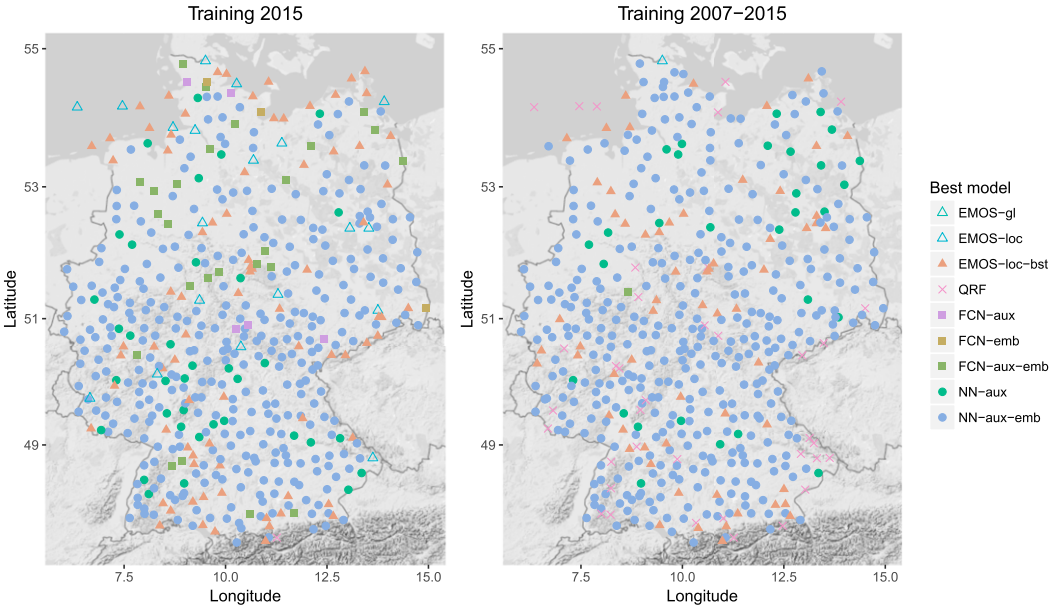


FIG. 4. Observation station locations color coded by the best performing model (in terms of mean CRPS over calendar year 2016) for models trained on data from (left) 2015 and (right) 2007 to 2015. Point shapes indicate the type of model.

Additionally, we evaluated the statistical significance of the differences between the competing postprocessing methods using a combination of Diebold–Mariano tests (Diebold and Mariano 1995) and a Benjamini and Hochberg (1995) procedure to account for temporal and spatial dependencies of forecast errors. We thereby follow the suggestions of Wilks (2016); the mathematical details are deferred to the appendix. The results (provided in the supplemental material) generally indicate high ratios of stations with significant score differences in favor of the neural network models. Even when compared to the second-best-performing model, EMOS-loc-bst, NN-aux-emb is significantly better at 30% of the stations and worse at only 2% or less for both training periods.

### c. *Computational aspects*

While a direct comparison of computation times for the different methods is difficult, even the most complex network methods are a factor of 2 or more faster than EMOS-loc-bst. This includes creating an ensemble of 10 different model realizations. QRF is by far the slowest method, being roughly 10 times slower than EMOS-loc-bst. Complex neural networks benefit substantially from running on a graphics processing unit (GPU) compared to running on the core processing unit (CPU; roughly 6 times slower for NN-aux-emb). Neural network–ready GPUs are now widely available in many scientific computing environments or via cloud computing.[8] For more details on the computational methods and results see the supplemental material.

## 6. Feature importance

To assess the relative importance of all features, we use a technique called permutation importance that was first described within the context of random forests (Breiman 2001). We randomly shuffle each predictor/feature in the validation set one at a time and observe the increase in mean CRPS compared to the unpermuted features. While unable to capture colinearities between features, this method does not require reestimating the model with each individual feature omitted.

Consider a random permutation of station and time indices $\pi(s, t)$ and let $\mathbf{X}_{s,t}^{\text{perm}_v}$ denote the vector of predictors where variable $v$ is permuted according to $\pi$ (i.e., a vector with $j$th entry):

$$\mathbf{X}_{s,t}^{\text{perm}_v(j)} = \begin{cases} \mathbf{X}_{s,t}^{(j)}, & j \neq v \\ \mathbf{X}_{\pi(s,t)}^{(v)}, & j = v \end{cases} \quad \text{for} \quad j = 1, \ldots, p \, .$$

The importance of input feature $v$ is computed as the mean CRPS difference:

[8] For example, see https://colab.research.google.com/.

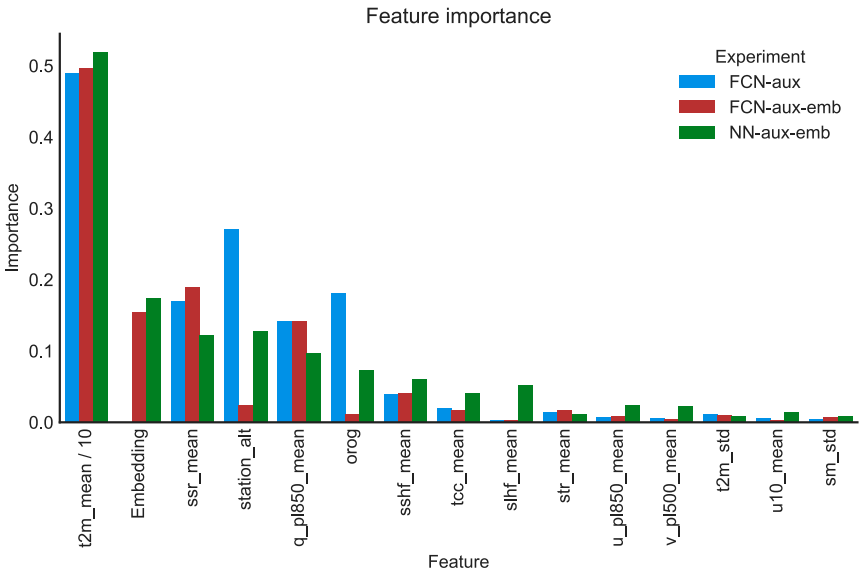


FIG. 5. Feature importance for the 15 most important predictors. Note that the values for t2m_mean are divided by 10. See Table 1 for variable abbreviations and descriptions.

$$\text{Importance}(v) = \frac{1}{ST}\sum_{s=1}^{S}\sum_{t=1}^{T}\left[\text{CRPS}(F|\mathbf{X}_{s,t}^{\text{perm}_v}, y_{s,t}) - \text{CRPS}(F|\mathbf{X}_{s,t}, y_{s,t})\right],$$

where we average over the entire evaluation set and $F|\mathbf{X}$ denotes the conditional forecast distribution given a vector of predictors.

We picked three network setups to investigate how feature importance changes by adding station embeddings and a nonlinear layer (Fig. 5). For the linear model without station embeddings (FCN-aux), the station altitude and orography, the altitude of the model grid cell, are the most important predictors after the mean temperature forecast. This makes sense since our interpolation from the forecast model grid to the station does not adjust for the height of the surface station. The only other features with significant importance are the mean shortwave radiation flux and the 850-hPa specific humidity. Adding station embeddings (FCN-aux-emb) reduces the significance of the station altitude information, which now seems to be encoded in the latent embedding features. The nonlinearity added by the hidden layer in NN-aux-emb increases the sensitivity to permuting input features overall and distributes the feature importance more evenly. In particular, we note an increase in the importance of the station altitude and orography but also the sensible and latent heat flux and total cloud cover.

The most important features, apart from the obvious mean forecast temperature and station altitude, seem to be indicative of insolation, either directly like the shortwave flux or indirectly like the 850-hPa humidity. It is interesting that the latter seems to be picked by the algorithms as a proxy for cloud cover rather than the direct cloud cover feature, potentially due to a lack of forecast skill of the total cloud cover predictions (e.g., Hemri et al. 2016). Curiously, the temperature standard deviation is not an important feature for the postprocessing models. We suspect that this is a consequence of the low correlation between the raw ensemble standard deviation and the forecast error ($r = 0.15$ on the test set) and the general underdispersion (mean spread–error ratio of 0.51). The postprocessing algorithms almost double the spread to achieve a spread–error ratio of 0.95. The correlation of the raw and postprocessed ensemble spreads is 0.39. suggesting that the postprocessing is mostly an additive correction to the ensemble spread.

Note that this method of assessing feature importance is in principle possible for boosting- and QRF-based models. However, for the local implementations of the algorithm the importance changes from station to station, making interpretation more difficult.

## 7. Discussion

Here, we discuss some approaches we attempted that failed to improve our results, as well as directions for future research.

Having to describe the distribution of the target variable in parametric techniques is a nontrivial task. For temperature, a Gaussian distribution is a good approximation but for other variables, such as wind speed or precipitation, finding a distribution that fits the data is a substantial challenge (e.g., Taillardat et al. 2016;

Baran and Lerch 2018). Ideally, a machine learning algorithm would learn to predict the full probability distribution rather than distribution parameters only. One way to achieve this is to approximate the forecast distribution by a combination of uniform distributions and predicting the probability of the temperature being within prespecified bins. Initial experiments indicate that the neural network is able to produce a good approximation of a Gaussian distribution but the skill was comparable only to the raw ensemble. This suggests that for target variables that are well approximated by a parametric distribution, utilizing these distributions is advantageous. One direction for future research is to apply this approach to more complex variables.

Standard EMOS models are often estimated based on so-called rolling training windows with data from previous days only in order to incorporate temporal dependencies of ensemble forecast errors. For neural networks, one way to incorporate temporal dependencies is to use convolutional or recurrent neural networks (Schmidhuber 2015) which can proces sequences as an input. In our tests, this leads to more overfitting without an improvement in the validation score. For other datasets, however, we believe that these approaches are worth revisiting. Temporal dependencies of forecast errors might further include seasonal effects. For standard EMOS models, it is possible to account for seasonality by estimating the model based on a centered window $[d_0 - m, d_0 + m]$ around the current day $d_0$. For the local EMOS model this resulted in negligible improvements only. For postprocessing models with additional predictors seasonal effects can, for example, be included by considering the month of $d_0$ as an input feature.

One popular way to combat overfitting in machine learning algorithms is through data augmentation. In the example of image recognition models, the training images are randomly rotated, flipped, zoomed, etc. to artificially increase the sample size (e.g., Krizhevsky et al. 2012). We tried a similar approach by adding random noise of a reasonable scale to the input features, but found no improvement in the validation score. A potential alternative to adding random noise might be augmenting the forecasts for a station with data from neighboring stations or grid points.

Similarly to rolling training windows for the traditional EMOS models, we tried updating the neural network each day during the validation period with the data from the previous time step, but found no improvements. This supports our observation that rolling training windows only bring marginal improvements for the benchmark EMOS models. Such an online learning approach could be more relevant in an operational setting, however, where model versions might change frequently or it is too expensive to reestimate the entire postprocessing model every time new data become available.

We have restricted the set of predictors to observation station characteristics and summary statistics (mean and standard deviation) of ensemble predictions of several weather variables. Recently, flexible distribution-to-distribution regression network models have been proposed in the machine learning literature (e.g., Oliva et al. 2013; Kou et al. 2018). Adaptations of such approaches might enable the use of the entire ensemble forecast of each predictor variable as an input feature. However, training of these substantially more complex models likely requires longer training periods than were possible in our study.

Another possible extension would be to postprocess forecasts on the entire two-dimensional grid, rather than individual stations locations, for example, by using convolutional neural networks. This adds computational complexity and probably requires more training data but could provide information about the large-scale weather patterns and help to produce spatially consistent predictions.

We have considered probabilistic forecasts of a single weather variable at a single location and look-ahead time only. However, many applications require accurate models of cross-variable, spatial, and temporal dependence structures, and much recent work has been focused on multivariate postprocessing methods (e.g., Schefzik et al. 2013). Extending the neural network–based approaches to multivariate forecast distributions accounting for such dependencies presents a promising starting point for future research.

## 8. Conclusions

In this study we demonstrated how neural networks can be used for distributional regression postprocessing of ensemble weather forecasts. Our neural network models significantly outperform state-of-the-art postprocessing techniques while being computationally more efficient. The main advantages of using neural networks are the ability to capture nonlinear relations between arbitrary predictors and distribution parameters without having to specify appropriate link functions, and the ease of adding station information into a global model by using embeddings. The network model parameters are estimated by optimizing the CRPS, a nonstandard choice in the machine learning literature tailored to probabilistic forecasting. Furthermore, the rapid pace of development in the deep learning community provides flexible and efficient modeling techniques and software libraries. The presented approach can therefore be easily applied to other problems.

The building blocks of our network model architecture provide general insight into the relative importance of model properties for postprocessing ensemble forecasts. Specifically, the results indicate that encoding local information is very important for providing skillful probabilistic temperature forecasts. Further, including covariate information via auxiliary variables improves the results considerably, particularly when allowing for nonlinear relations of predictors and forecast distribution parameters. Ideally, any postprocessing model should thus strive to incorporate all of these aspects.

We also showed that a trained machine learning model can be used to gain meteorological insight. In our case, it allowed us to identify the variables that are most important for correcting systematic temperature forecast errors of the ensemble. Within this context, neural networks are somewhat interpretable and give us more information than we originally asked for. While a direct interpretation of the individual parameters of the model is intractable, this challenges the common notion of neural networks as pure black boxes.

Because of their flexibility, neural networks are ideally suited to handle the increasing amounts of model and observation data as well as the diverse requirements for correcting multifaceted aspects of systematic ensemble forecast errors. We anticipate, therefore, that they will provide a valuable addition to the modeler's toolkit for many areas of statistical postprocessing and forecasting.

*Acknowledgments.* The research leading to these results has been done within the subprojects A6 "Representing forecast uncertainty using stochastic physical parameterizations" and C7 "Statistical postprocessing and stochastic physics for ensemble predictions" of the Transregional Collaborative Research Center SFB/TRR 165 "Waves to Weather" funded by the German Research Foundation (DFG). SL is grateful for infrastructural support by the Klaus Tschira Foundation. The authors thank Tilmann Gneiting, Alexander Jordan, and Maxime Taillardat for helpful discussions and for providing code. The initial impetus for this work stems from a meeting with Kai Polsterer, who presented a probabilistic neural network–based approach to astrophysical image data analysis. We are grateful to Jakob Messner and two anonymous referees for constructive comments on an earlier version of the manuscript.

## APPENDIX

### Forecast Evaluation

For the purpose of this appendix, we denote a generic probabilistic forecast for 2-m temperature $y_{s,t}$ at station $s$ and time $t$ by $F_{s,t}$. Note that $F_{s,t}$ may be a parametric forecast distribution represented by CDF or a probability density function (PDF), an ensemble forecast $x_{s,t}^{\mathrm{t2m},1}, \ldots, x_{s,t}^{\mathrm{t2m},50}$, or a set of quantiles. We may choose to suppress the index $s$, $t$ at times for ease of notation.

#### a. Calibration and sharpness

As argued by Gneiting et al. (2007), probabilistic forecasts should generally aim to maximize sharpness subject to calibration. In a nutshell, a forecast is called calibrated if the realizing observation cannot be distinguished from a random draw from the forecast distribution. Calibration thus refers to the statistical consistency between forecast distribution and observation. By contrast, sharpness is a property of the forecast only and refers to the concentration of the predictive distribution. The calibration of ensemble forecasts can be assessed via verification rank (VR) histograms summarizing the distribution of ranks of the observation $y_{s,t}$ when it is pooled with the ensemble forecast (Hamill 2001; Gneiting et al. 2007; Wilks 2011). For continuous forecast distributions, histograms of the PIT $F_{s,t}(y_{s,t})$ provide analogs of verification rank histograms. Calibrated forecasts result in uniform VR and PIT histograms, and deviations from uniformity indicate specific systematic errors such as biases or an underrepresentation of the forecast uncertainty.

#### b. Proper scoring rules

For comparative model assessment, proper scoring rules allow simultaneous evaluation of calibration and sharpness (Gneiting and Raftery 2007). A scoring rule $S(F, y)$ assigns a numerical score to a pair of probabilistic forecasts $F$ and corresponding realizing observations $y$, and is called proper relative to a class of forecast distributions $\mathscr{F}$ if

$$\mathbb{E}_{Y\sim G}S(G,Y) \le \mathbb{E}_{Y\sim G}S(F,Y) \quad \text{for all} \quad F, G \in \mathscr{F},$$

that is, if the expected score is optimized if the true distribution of the observation is issued as forecast. Here, scoring rules are considered to be negatively oriented, with smaller scores indicating better forecasts

Popular examples of proper scoring rules include the logarithmic score (LogS; Good 1952):

$$\mathrm{LogS}(F,y) = -\log[f(y)],$$

where $y$ denotes the observations and $f$ denotes the PDF of the forecast distribution and the continuous ranked probability score (CRPS; Matheson and Winkler 1976):

$$\mathrm{CRPS}(F,y) = \int_{-\infty}^{\infty} [F(z) - 1(y \le z)]^2 \, dz, \quad \text{(A1)}$$

where $F$ denotes the CDF of the forecast distribution with finite first moment and $1(y \le z)$ is an indicator

function that is 1 if $y \le z$ and 0 otherwise. The integral in (A1) can be computed analytically for ensemble forecasts and a variety of continuous forecast distributions (see, e.g., Jordan et al. 2018). Specifically, the CRPS of a Gaussian distribution with mean value $\mu$ and standard deviation $\sigma$ can be computed as

$$\mathrm{CRPS}(F_{\mu,\sigma}, y) = \sigma\left\{\frac{y-\mu}{\sigma}\left[2\Phi\left(\frac{y-\mu}{\sigma}\right) - 1\right] + 2\varphi\left(\frac{y-\mu}{\sigma}\right) - \frac{1}{\sqrt{\pi}}\right\}, \quad \text{(A2)}$$

where $\Phi$ and $\varphi$ denote the CDF and PDF of a standard Gaussian distribution, respectively (Gneiting et al. 2005).

Apart from forecast evaluation, proper scoring rules can also be used for parameter estimation. Following the generic optimum score estimation framework of Gneiting and Raftery (2007, section 9.1), the parameters of a forecast distribution are determined by optimizing the value of a proper scoring rule, on average over a training sample. Optimum score estimation based on the LogS then corresponds to classical maximum likelihood estimation, whereas optimum score estimation based on the CRPS is often employed as a more robust alternative in meteorological applications. Analytical closed-form solutions of the CRPS, for example for a Gaussian distribution in (A2), allow for computing analytical gradient functions that can be leveraged in numerical optimization; see Jordan et al. (2018) for details.

In practical applications, scoring rules are usually computed as averages over stations and/or time periods. To assess the relative improvement over a reference forecast $F_{\mathrm{ref}}$, we further introduce the continuous ranked probability skill score:

$$\mathrm{CRPSS}(F, y) = 1 - \frac{\mathrm{CRPS}(F, y)}{\mathrm{CRPS}(F_{\mathrm{ref}}, y)},$$

which is positively oriented and can be interpreted as a relative improvement over the reference. The CRPSS is usually computed as the skill score of the CRPS averages.

### c. *Statistical tests of equal predictive performance*

Formal statistical tests of equal forecast performance for assessing statistical significance of score differences have been widely used in the economic literature. Consider two forecasts, $F^1$ and $F^2$, with corresponding mean scores $\overline{S}(F^i) = 1/n\sum_{j=1}^{n} S(F_j^i, y_j)$ for $i = 1, 2$ over a test $j = 1, \ldots, n$, where we assume that the forecast $F_j^i$ was issued $k$ time steps before the observation $y_j$ was recorded. Diebold and Mariano (1995) propose the test statistic

$$t_n = \sqrt{n}\,\frac{\overline{S}(F^1) - \overline{S}(F^2)}{\hat{\sigma}_n},$$

where $\hat{\sigma}_n$ is an estimator of the asymptotic standard deviation of the score difference between $F^1$ and $F^2$. Under standard regularity conditions, $t_n$ asymptotically follows a standard normal distribution under the null hypothesis of equal predictive performance of $F^1$ and $F^2$. Thereby, negative values of $t_n$ indicate superior predictive performance of $F^1$, whereas positive values indicate superior performance of $F^2$. To account for temporal dependencies in the score differences, we use the square root of the sample autocovariance up to lag $k-1$ as estimator $\hat{\sigma}_n$ following Diebold and Mariano (1995). We employ Diebold–Mariano tests on an observation station level; that is, the mean CRPS values are determined by averaging over all scores at the specific station $s_0 \in \{1, \ldots, S\}$ of interest:

$$\overline{\mathrm{CRPS}}(F_{s_0}^i) = \frac{1}{T}\sum_{t=1}^{T} F_{s_0,t}^i,$$

where $t = 1, \ldots, T$ denotes days in the evaluation period.

Compared to previous uses of Diebold–Mariano tests in postprocessing applications (e.g., Baran and Lerch 2016), we further account for spatial dependencies of score differences at the different stations. Following the suggestions of Wilks (2016), we apply a Benjamini and Hochberg (1995) procedure to control the false discovery rate at level $\alpha = 0.05$. In a nutshell, the algorithm requires a higher standard in order to reject a local null hypothesis of equal predictive performance by selecting a threshold $p$ value ($p^*$) based on the set of ordered local $p$ values: $p_{(1)}, \ldots, p_{(S)}$. Particularly, $p^*$ is the largest $p_{(i)}$ that is not larger than $i/S \times \alpha$, where $S$ is the number of tests (i.e., the number of stations in the evaluation set).

## REFERENCES


Abadi, M., and Coauthors, 2016: Tensorflow: A system for large-scale machine learning. *Proc. USENIX 12th Symp. on Operating Systems Design and Implementation*, Savannah, GA, Advanced Computing Systems Association, 265–283, https://www.usenix.org/system/files/conference/osdi16/osdi16-abadi.pdf.

Angermueller, C., T. Pärnamaa, L. Parts, and O. Stegle, 2016: Deep learning for computational biology. *Mol. Syst. Biol.*, **12**, 878, https://doi.org/10.15252/msb.20156651.

Athey, S., J. Tibshirani, and S. Wager, 2016: Generalized random forests. arXiv.org, https://arxiv.org/abs/1610.01271.

Baran, S., and S. Lerch, 2015: Log-normal distribution based ensemble model output statistics models for probabilistic wind-speed forecasting. *Quart. J. Roy. Meteor. Soc.*, **141**, 2289–2299, https://doi.org/10.1002/qj.2521.

——, and ——, 2016: Mixture EMOS model for calibrating ensemble forecasts of wind speed. *Environmetrics*, **27**, 116–130, https://doi.org/10.1002/env.2380.

——, and ——, 2018: Combining predictive distributions for the statistical post-processing of ensemble forecasts. *Int. J. Forecasting*, **34**, 477–496, https://doi.org/10.1016/j.ijforecast.2018.01.005.

Bauer, P., A. Thorpe, and G. Brunet, 2015: The quiet revolution of numerical weather prediction. *Nature*, **525**, 47–55, https://doi.org/10.1038/nature14956.

Benjamini, Y., and Y. Hochberg, 1995: Controlling the false discovery rate: A practical and powerful approach to multiple testing. *J. Roy. Stat. Soc.*, **57B**, 289–300.

Bougeault, P., and Coauthors, 2010: The THORPEX Interactive Grand Global Ensemble. *Bull. Amer. Meteor. Soc.*, **91**, 1059–1072, https://doi.org/10.1175/2010BAMS2853.1.

Breiman, L., 2001: Random forests. *Mach. Learn.*, **45**, 5–32, https://doi.org/10.1023/A:1010933404324.

——, J. H. Friedman, R. A. Olshen, and C. J. Stone, 1984: *Classification and Regression Trees*. Wadsworth, 368 pp.

Chollet, F., and Coauthors, 2015: Keras: The Python Deep Learning library. https://keras.io.

Chu, Y., H. T. C. Pedro, and C. F. M. Coimbra, 2013: Hybrid intra-hour DNI forecasts with sky image processing enhanced by stochastic learning. *Sol. Energy*, **98**, 592–603, https://doi.org/10.1016/j.solener.2013.10.020.

Diebold, F. X., and R. S. Mariano, 1995: Comparing predictive accuracy. *J. Bus. Econ. Stat.*, **13**, 253–263, https://doi.org/10.1080/07350015.1995.10524599.

D'Isanto, A., and K. L. Polsterer, 2018: Photometric redshift estimation via deep learning-generalized and pre-classification-less, image based, fully probabilistic redshifts. *Astron. Astrophys.*, **609**, A111, https://doi.org/10.1051/0004-6361/201731326.

Gagne, D. J., A. McGovern, and M. Xue, 2014: Machine learning enhancement of storm-scale ensemble probabilistic quantitative precipitation forecasts. *Wea. Forecasting*, **29**, 1024–1043, https://doi.org/10.1175/WAF-D-13-00108.1.

——, ——, S. E. Haupt, R. A. Sobash, J. K. Williams, and M. Xue, 2017: Storm-based probabilistic hail forecasting with machine learning applied to convection-allowing ensembles. *Wea. Forecasting*, **32**, 1819–1840, https://doi.org/10.1175/WAF-D-17-0010.1.

Gebetsberger, M., J. W. Messner, G. J. Mayr, and A. Zeileis, 2017: Estimation methods for non-homogeneous regression models: Minimum continuous ranked probability score vs. maximum likelihood. Faculty of Economics and Statistics Working Paper 2017-23, University of Innsbruck, 21 pp., https://ideas.repec.org/p/inn/wpaper/2017-23.html.

——, R. Stauffer, G. J. Mayr, and A. Zeileis, 2018: Skewed logistic distribution for statistical temperature post-processing in mountainous areas. Faculty of Economics and Statistics Working Paper 2018-06, University of Innsbruck, 16 pp., https://ideas.repec.org/p/inn/wpaper/2018-06.html.

Gentine, P., M. S. Pritchard, S. Rasp, G. Reinaudi, and G. Yacalis, 2018: Could machine learning break the convection parameterization deadlock? *Geophys. Res. Lett.*, **45**, 5742–5751, https://doi.org/10.1029/2018GL078202.

Gneiting, T., and A. E. Raftery, 2007: Strictly proper scoring rules, prediction, and estimation. *J. Amer. Stat. Assoc.*, **102**, 359–378, https://doi.org/10.1198/016214506000001437.

——, ——, A. H. Westveld, and T. Goldman, 2005: Calibrated probabilistic forecasting using ensemble model output statistics and minimum CRPS estimation. *Mon. Wea. Rev.*, **133**, 1098–1118, https://doi.org/10.1175/MWR2904.1.

——, F. Balabdaoui, and A. E. Raftery, 2007: Probabilistic forecasts, calibration and sharpness. *J. Roy. Stat. Soc.*, **69B**, 243–268, https://doi.org/10.1111/j.1467-9868.2007.00587.x.

Goh, G. B., N. O. Hodas, and A. Vishnu, 2017: Deep learning for computational chemistry. *J. Comput. Chem.*, **38**, 1291–1307, https://doi.org/10.1002/jcc.24764.

Good, I. J., 1952: Rational decisions. *J. Roy. Stat. Soc.*, **14B**, 107–114.

Goodfellow, I., Y. Bengio, and A. Courville, 2016: *Deep Learning*. MIT Press, 775 pp.

Guo, C., and F. Berkhahn, 2016: Entity embeddings of categorical variables. arXiv.org, https://arxiv.org/abs/1604.06737.

Hamill, T. M., 2001: Interpretation of rank histograms for verifying ensemble forecasts. *Mon. Wea. Rev.*, **129**, 550–560, https://doi.org/10.1175/1520-0493(2001)129<0550:IORHFV>2.0.CO;2.

Hemri, S., M. Scheuerer, F. Pappenberger, K. Bogner, and T. Haiden, 2014: Trends in the predictive performance of raw ensemble weather forecasts. *Geophys. Res. Lett.*, **41**, 9197–9205, https://doi.org/10.1002/2014GL062472.

——, T. Haiden, and F. Pappenberger, 2016: Discrete postprocessing of total cloud cover ensemble forecasts. *Mon. Wea. Rev.*, **144**, 2565–2577, https://doi.org/10.1175/MWR-D-15-0426.1.

Jordan, A., F. Krüger, and S. Lerch, 2018: Evaluating probabilistic forecasts with scoringRules. arXiv.org, https://arxiv.org/abs/1709.04743.

Junk, C., L. Delle Monache, and S. Alessandrini, 2015: Analog-based ensemble model output statistics. *Mon. Wea. Rev.*, **143**, 2909–2917, https://doi.org/10.1175/MWR-D-15-0095.1.

Kahle, D., and H. Wickham, 2013: Ggmap: Spatial visualization with ggplot2. *R J.*, **5**, 144–161.

Kingma, D. P., and J. Ba, 2014: Adam: A method for stochastic optimization. arXiv.org, https://arxiv.org/abs/1412.6980.

Kou, C., H. K. Lee, and T. K. Ng, 2018: Distribution regression network. arXiv.org, https://arxiv.org/abs/1804.04775.

Krizhevsky, A., I. Sutskever, and G. E. Hinton, 2012: Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems 25*, F. Pereira et al., Eds., Curran Associates, 1097–1105.

Krüger, F., S. Lerch, T. L. Thorarinsdottir, and T. Gneiting, 2016: Probabilistic forecasting and comparative model assessment based on Markov chain Monte Carlo output. arXiv.org, https://arxiv.org/abs/1608.06802.

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

LeCun, Y., Y. Bengio, and G. Hinton, 2015: Deep learning. *Nature*, **521**, 436–444, https://doi.org/10.1038/nature14539.

Lerch, S., and T. L. Thorarinsdottir, 2013: Comparison of non-homogeneous regression models for probabilistic wind speed forecasting. *Tellus*, **65A**, 21206, https://doi.org/10.3402/tellusa.v65i0.21206.

——, and S. Baran, 2017: Similarity-based semilocal estimation of post-processing models. *J. Roy. Stat. Soc.*, **66C**, 29–51, https://doi.org/10.1111/rssc.12153.

Liu, Y., E. Racah, J. Correa, A. Khosrowshahi, D. Lavers, K. Kunkel, M. Wehner, and W. Collins, 2016: Application of deep convolutional neural networks for detecting extreme weather in climate datasets. arXiv.org, https://arxiv.org/abs/1605.01156.

Matheson, J. E., and R. L. Winkler, 1976: Scoring rules for continuous probability distributions. *Manage. Sci.*, **22**, 1087–1096, https://doi.org/10.1287/mnsc.22.10.1087.

McGovern, A., K. L. Elmore, D. J. Gagne, S. E. Haupt, C. D. Karstens, R. Lagerquist, T. Smith, and J. K. Williams, 2017: Using artificial intelligence to improve real-time decision-making for

high-impact weather. *Bull. Amer. Meteor. Soc.*, **98**, 2073–2090, https://doi.org/10.1175/BAMS-D-16-0123.1.

Meinshausen, N., 2006: Quantile regression forests. *J. Mach. Learn. Res.*, **7**, 983–999.

——, 2017: QuantregForest: Quantile regression forests. R Package version 1.3-7, https://CRAN.R-project.org/package=quantregForest.

Messner, J. W., G. J. Mayr, D. S. Wilks, and A. Zeileis, 2014: Extending extended logistic regression: Extended versus separate versus ordered versus censored. *Mon. Wea. Rev.*, **142**, 3003–3014, https://doi.org/10.1175/MWR-D-13-00355.1.

——, ——, and A. Zeileis, 2016: Heteroscedastic censored and truncated regression with crch. *R J.*, **8**, 173–181.

——, ——, and ——, 2017: Nonhomogeneous boosting for predictor selection in ensemble postprocessing. *Mon. Wea. Rev.*, **145**, 137–147, https://doi.org/10.1175/MWR-D-16-0088.1.

Nielsen, M. A., 2015: *Neural Networks and Deep Learning*. Determination Press, http://neuralnetworksanddeeplearning.com/.

Oliva, J., B. Póczos, and J. Schneider, 2013: Distribution to distribution regression. *Proc. 30th Int. Conf. on Machine Learning*, Atlanta, GA, Association for Computing Machinery, 1049–1057.

Python Software Foundation, 2017: Python software, version 3.6.4. Python Software Foundation, https://www.python.org/.

Raftery, A. E., T. Gneiting, F. Balabdaoui, and M. Polakowski, 2005: Using Bayesian model averaging to calibrate forecast ensembles. *Mon. Wea. Rev.*, **133**, 1155–1174, https://doi.org/10.1175/MWR2906.1.

Rasp, S., M. S. Pritchard, and P. Gentine, 2018: Deep learning to represent subgrid processes in climate models. *Proc. Natl. Acad. Sci. USA*, **115**, 9684–9689, https://doi.org/10.1073/pnas.1810286115.

R Core Team, 2017: R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria, https://www.R-project.org/.

Schefzik, R., T. L. Thorarinsdottir, and T. Gneiting, 2013: Uncertainty quantification in complex simulation models using ensemble copula coupling. *Stat. Sci.*, **28**, 616–640, https://doi.org/10.1214/13-STS443.

Scheuerer, M., 2014: Probabilistic quantitative precipitation forecasting using ensemble model output statistics. *Quart. J. Roy. Meteor. Soc.*, **140**, 1086–1096, https://doi.org/10.1002/qj.2183.

——, and T. M. Hamill, 2015: Statistical postprocessing of ensemble precipitation forecasts by fitting censored, shifted gamma distributions. *Mon. Wea. Rev.*, **143**, 4578–4596, https://doi.org/10.1175/MWR-D-15-0061.1.

——, and D. Möller, 2015: Probabilistic wind speed forecasting on a grid based on ensemble model output statistics. *Ann. Appl. Stat.*, **9**, 1328–1349, https://doi.org/10.1214/15-AOAS843.

Schlosser, L., T. Hothorn, R. Stauffer, and A. Zeileis, 2018: Distributional regression forests for probabilistic precipitation forecasting in complex terrain. arXiv.org, https://arxiv.org/abs/1804.02921.

Schmidhuber, J., 2015: Deep learning in neural networks: An overview. *Neural Networks*, **61**, 85–117, https://doi.org/10.1016/j.neunet.2014.09.003.

Taillardat, M., O. Mestre, M. Zamo, and P. Naveau, 2016: Calibrated ensemble forecasts using quantile regression forests and ensemble model output statistics. *Mon. Wea. Rev.*, **144**, 2375–2393, https://doi.org/10.1175/MWR-D-15-0260.1.

——, A.-L. Fougères, P. Naveau, and O. Mestre, 2017: Forest-based methods and ensemble model output statistics for rainfall ensemble forecasting. arXiv.org, https://arxiv.org/abs/1711.10937.

Taylor, J. W., 2000: A quantile regression neural network approach to estimating the conditional density of multiperiod returns. *J. Forecasting*, **19**, 299–311, https://doi.org/10.1002/1099-131X(200007)19:4<299::AID-FOR775>3.0.CO;2-V.

Thorarinsdottir, T. L., and T. Gneiting, 2010: Probabilistic forecasts of wind speed: Ensemble model output statistics by using heteroscedastic censored regression. *J. Roy. Stat. Soc.*, **173A**, 371–388, https://doi.org/10.1111/j.1467-985X.2009.00616.x.

Wang, F., Z. Mi, S. Su, and H. Zhao, 2012: Short-term solar irradiance forecasting model based on artificial neural network using statistical feature parameters. *Energies*, **5**, 1355–1370, https://doi.org/10.3390/en5051355.

Wilks, D. S., 2011: *Statistical Methods in the Atmospheric Sciences*. 3rd ed. Elsevier, 676 pp.

——, 2016: "The stippling shows statistically significant grid points": How research results are routinely overstated and overinterpreted, and what to do about it. *Bull. Amer. Meteor. Soc.*, **97**, 2263–2273, https://doi.org/10.1175/BAMS-D-15-00267.1.