Statistical Postprocessing of Wind Speed Forecasts Using Convolutional Neural Networks

SIMON VELDKAMP,[a,b] KIRIEN WHAN,[a] SJOERD DIRKSEN,[b] AND MAURICE SCHMEITS[a]

[a] *Royal Netherlands Meteorological Institute (KNMI), De Bilt, Netherlands*
[b] *Mathematical Institute, Utrecht University, Utrecht, Netherlands*

(Manuscript received 6 July 2020, in final form 6 January 2021)

ABSTRACT: Current statistical postprocessing methods for probabilistic weather forecasting are not capable of using full spatial patterns from the numerical weather prediction (NWP) model. In this paper, we incorporate spatial wind speed information by using convolutional neural networks (CNNs) and obtain probabilistic wind speed forecasts in the Netherlands for 48 h ahead, based on KNMI's deterministic HARMONIE-AROME NWP model. The probabilistic forecasts from the CNNs are shown to have higher Brier skill scores for medium to higher wind speeds, as well as a better continuous ranked probability score (CRPS) and logarithmic score, than the forecasts from fully connected neural networks and quantile regression forests. As a secondary result, we have compared the CNNs using three different density estimation methods [quantized softmax (QS), kernel mixture networks, and fitting a truncated normal distribution], and found the probabilistic forecasts based on the QS method to be best.

KEYWORDS: Forecast verification/skill; Probability forecasts/models/distribution; Model output statistics; Deep learning; Machine learning; Neural networks

1. Introduction

Accurate and reliable weather forecasts are important in many branches of society. Decision-making in, for example, agriculture, aviation, and renewable energy production are all dependent on skillful weather forecasts (e.g., Wilczak et al. 2015). Furthermore, extreme weather can be dangerous for life and property, and it is therefore important to give reliable warnings when dangerous weather can be expected. Extreme winds have a large impact in the Netherlands. Two western European wind storms caused one insurance group to pay an estimated €100 million (EUR) due to damages to individuals, businesses, and the agricultural sector (https://news.achmea.nl/achmea-pays-out-over-eur-100-million-to-customers-hit-by-january-storms/). KNMI issued eight code orange weather warnings in 2018, of which at least three were associated with extreme wind speeds, including the two winter storms that caused the damage noted above [https://www.knmi.nl/kennis-en-datacentrum/uitleg/archief-code-oranje-rood-in-2018 (in Dutch)].

Forecasts are generally produced by numerical weather prediction (NWP) models, such as the HARMONIE-AROME model (Bengtsson et al. 2017) of the Royal Netherlands Meteorological Institute (KNMI). To make computation of NWP models feasible it is necessary to make simplifying assumptions, but the resulting parameterization of the subgrid-scale processes can introduce errors in the forecast. In addition, a perfect initialization of these models is not possible. As the atmosphere is a famously chaotic system (Lorenz 1963), every forecast is therefore inherently uncertain. A single forecast given by an NWP model does not provide an estimate of this uncertainty, even though such an estimate is important for decision-makers.

Forecast uncertainty is usually estimated from an ensemble of predictions where each member is the outcome of an NWP model run with a perturbed initial state and/or perturbed physical parameterizations. This approach is, however, computationally expensive and the results are often still biased and underdispersed (Gneiting and Raftery 2005).

To correct biases and systematic errors in the ensemble spread, and to make probabilistic forecasts from deterministic NWP forecasts, one can use statistical postprocessing, based on past observations. A popular framework for statistical postprocessing is model output statistics (MOS; Glahn and Lowry 1972). In MOS a statistical relationship is derived between the forecasts provided by the NWP model and the corresponding observed measurements. In this way we can correct the bias and estimate the uncertainty in the forecast, based on deterministic or ensemble NWP model output and potentially some additional variables, such as the time of the year.

It is common practice in statistical postprocessing to fit a parametric distribution based on measurements and a set of potential predictors. These predictors can come from an ensemble forecast, as is the case in ensemble model output statistics (EMOS; Gneiting and Raftery 2005), or the predictors can come from a deterministic forecast in an MOS application (e.g., Whan and Schmeits 2018; Bakker et al. 2019). In our study we have used predictors from deterministic HARMONIE-AROME model output as a long HARMONIE-AROME ensemble dataset was not yet available. The quality of the fit is measured in terms of skill scores associated with scoring rules such as the continuous ranked probability score (CRPS; Matheson and Winkler 1976; Hersbach 2000; Gneiting and Raftery 2007). Wind speed has been modeled with a parametric distribution in an EMOS framework in Scheuerer and Möller (2015), Thorarinsdottir and Gneiting (2010), and Baran and Lerch (2016), where they used truncated normal and lognormal distributions. Furthermore in Lerch and Thorarinsdottir (2013) a mixture of truncated normal

Corresponding author: Maurice Schmeits, maurice.schmeits@knmi.nl

DOI: 10.1175/MWR-D-20-0219.1

and the generalized extreme value distribution was used with success. Ioannidis et al. (2018) tested a variety of distributions for wind speed in Denmark and found that the truncated normal distribution was the most skillful.

Parametric methods (i.e., EMOS) have been compared to quantile regression forests [QRF; Meinshausen (2006), a nonparametric technique based on random forests] for both wind speed and temperature forecasts in Taillardat et al. (2016), where QRF was found to be more skillful. QRF was also used in Whan and Schmeits (2018) for postprocessing of precipitation forecasts, and Rasp and Lerch (2018) for postprocessing 2m temperature forecasts. Rasp and Lerch (2018) also used fully connected neural networks (NNs) to determine the parameters of a normal distribution. This approach was shown to be more skillful than fitting a normal distribution in an EMOS framework for the statistical postprocessing of temperature forecasts. In contrast to MOS/EMOS, QRF and neural networks are both capable of learning nonlinear relationships, while for MOS/EMOS nonlinear dependencies must be specifically coded.

The aforementioned methods (MOS/EMOS, QRF, and fully connected neural networks) are not well suited to use high-dimensional structured spatial data, although they can use spatial information in a crude way, for example, by taking statistics of the predictor in an area around the station location (van der Plas et al. 2017). As weather forecasts are spatial in nature, it could be beneficial to use postprocessing methods that are capable of dealing with this spatial information. In the recent literature on machine learning, convolutional neural networks (CNNs) have strongly advanced the state-of-the-art on learning tasks involving this type of information, e.g., in image classification and time series analysis (see, e.g., LeCun et al. 2015; Krizhevsky et al. 2012). CNNs can potentially be of great benefit in the geosciences (Reichstein et al. 2019) and have already been applied in in the meteorological domain. For example, Liu et al. (2016) used CNNs to detect extreme weather events in climate datasets, and Shi et al. (2017) used a mix between a convolutional and a recurrent network for nowcasting precipitation. Additionally, CNNs have been used to make statistical forecasts of frontal systems (Lagerquist et al. 2019), 500 hPa geopotential height anomalies (Weyn et al. 2019), the probability of large hail using features from an NWP model (Gagne II et al. 2019), and to estimate the uncertainty in weather forecasts based on the state of the atmosphere in the initialization of an NWP model (Scher and Messori 2018).

The above studies demonstrate the wide variety of problems for which CNNs have been used. However, to the best of our knowledge, CNNs have not yet been used for probabilistic forecasting of wind speed using statistical postprocessing. We expect that the capability of CNNs to analyze spatial information of weather forecasts could make them a very beneficial new tool for this purpose. Independently of our work, Scheuerer et al. (2020) very recently investigated CNNs for probabilistic forecasting of precipitation on the subseasonal time scale in California and found them to improve over state-of-the-art postprocessing methods, and Dupuy et al. (2020) demonstrated that CNNs are more skillful than traditional methods (QRF and logistic regression) in postprocessing cloud cover forecasts.

In this study we apply convolutional neural networks for the postprocessing of +48 h deterministic wind speed forecasts in the Netherlands. We compare three different methods for fitting a (conditional) probability distribution using CNNs: quantized softmax (Oord et al. 2016), kernel mixture networks (Ambrogioni et al. 2017), and fitting a truncated normal distribution (e.g., Thorarinsdottir and Gneiting 2010) whose parameters are determined by the network. Furthermore, we examine whether convolutional neural networks are more skillful than fully connected neural networks and QRF.

This paper is structured as follows. In section 2 we give a description of the data that has been used in this study and in section 3 we give a short description of quantile regressions forests and (convolutional) neural networks and detail the models used in this study. Section 4 contains the results and, finally, section 5 contains the conclusions and discussion.

2. Data

The input data are provided by HARMONIE-AROME cycle 40 (HA40) used by KNMI. HA40 is a nonhydrostatic model that is run on a 2.5 km × 2.5 km grid. We use deterministic HA40 forecasts that are initialized at 0000 UTC and are valid at a lead time of 48 h. The predictand data are the 10-min-average wind speed observations in the extended winter period (mid-October to mid-April), at 10 m above the ground, from 46 weather stations in the Netherlands, which are shown in Table 1. These measurements are provided as rounded to the nearest full meter per second. The data from all the stations are pooled in the training dataset, meaning that the model is trained for all stations at once, without providing station-specific information other than HA40 surface roughness.

Reforecast data for HA40 is available from 2015 to 2017 and operational HA40 forecasts from winter 2018–19 are also available. These data are split into two sets, as shown in Table 2. The first set (2015–17) is used for model selection and training. We use a threefold cross validation on this model selection set. The second set (2018–19) is an independent dataset used for testing the selected models.

In threefold cross validation we train every model three times on the model selection set, each time with a different fold left out. The latter is then used to make predictions in order to validate the model. The sets are chosen in this way to ensure that there is at least 6 months between the training, validation, and test sets. This is necessary to avoid temporal correlations between the different datasets.

In this study we use two sets of predictors. The first set contains the HA40 forecasts of a number of variables in the neighborhood of the station, see Table 3. The second set contains the wind speed forecast from HA40 for a larger area around this station, the exact size of which has been determined in the hyperparameter search. The first set is used in all the methods described. The second set is only used for convolutional neural networks (in combination with the first set).

The set with the neighborhood predictors we use in this study is based on previous research on postprocessing of wind

TABLE 1. Location of weather stations in the Netherlands.

Station No.	Longitude (°E)	Latitude (°N)	Name
209	4.518	52.465	IJMOND
215	4.437	52.141	VOORSCHOTEN
225	4.555	52.463	IJMUIDEN
235	4.781	52.928	DE KOOY
240	4.790	52.318	SCHIPHOL
242	4.921	53.241	VLIELAND
248	5.174	52.634	WIJDENES
249	4.979	52.644	BERKHOUT
251	5.346	53.392	HOORN (TERSCHELLING)
258	5.401	52.649	HOUTRIBDIJK
260	5.180	52.100	DE BILT
267	5.384	52.898	STAVOREN
269	5.520	52.458	LELYSTAD
270	5.752	53.224	LEEUWARDEN
273	5.888	52.703	MARKNESSE
275	5.873	52.056	DEELEN
277	6.200	53.413	LAUWERSOOG
278	6.259	52.435	HEINO
279	6.574	52.750	HOOGEVEEN
280	6.585	53.125	EELDE
283	6.657	52.069	HUPSEL
285	6.399	53.575	HUIBERTGAT
286	7.150	53.196	NIEUW BEERTA
290	6.891	52.274	TWENTHE
308	3.379	51.381	CADZAND
310	3.596	51.442	VLISSINGEN
312	3.622	51.768	OOSTERSCHELDE
313	3.242	51.505	VLAKTE V.D. RAAN
315	3.998	51.447	HANSWEERT
316	3.694	51.657	SCHAAR
319	3.861	51.226	WESTDORPE
323	3.884	51.527	WILHELMINADORP
324	4.006	51.596	STAVENISSE
330	4.122	51.992	HOEK VAN HOLLAND
331	4.193	51.480	THOLEN
340	4.342	51.449	WOENSDRECHT
343	4.313	51.893	R'DAM-GEULHAVEN
344	4.447	51.962	ROTTERDAM
348	4.926	51.970	CABAUW
350	4.936	51.566	GILZE-RIJEN
356	5.146	51.859	HERWIJNEN
370	5.377	51.451	EINDHOVEN
375	5.707	51.659	VOLKEL
377	5.763	51.198	ELL
380	5.762	50.906	MAASTRICHT
391	6.197	51.498	ARCEN

speed forecasts by Ioannidis et al. (2018) and Taillardat et al. (2016). Based on their results we take the variables shown in Table 3 as the set of potential predictors.

The grid point closest to the station is used for the surface roughness. For the other variables we pick a number of grid boxes around the station and determine the mean value, maximal value, and minimal value of each predictor in this region, so that the method receives some information about the spatial uncertainty in the weather forecast.

TABLE 2. Definition of the different subsets used in cross validation and testing.

Model selection	Fold 1	Oct–Dec 2015 and Jan–Mar 2016
	Fold 2	Oct–Dec 2016 and Jan–Mar 2017
	Fold 3	Oct–Dec 2017 and Jan–Mar 2015
Test set		Nov–Dec 2018, Jan–Mar 2019, and Oct–Nov 2019

For the surface roughness only the closest grid point has been taken since it should reflect the conditions in the direct neighborhood of the station as closely as possible. Which predictors we will use, the number of grid boxes used, and whether to take the mean, maximum, minimum or a combination of them is decided for every method independently in the hyperparameter search, which is described in section 3e.

3. Methods

Three different methods are compared in this study: quantile regression forests, fully connected neural networks, and convolutional neural networks. We also compare three different methods for conditional density estimation using convolutional neural networks. Some of the models are trained by using the errors of linear regression as target variables instead of the observed measurements. We will motivate this choice for the various models below.

a. *Quantile regression forests*

Quantile regression forests (Meinshausen 2006) is a nonparametric method for estimating quantiles and, more generally, a conditional cumulative distribution function. The algorithm is based on random forests (Breiman 2001). Whereas a trained random forest outputs a point prediction by taking the average of the terminal nodes, QRF returns an estimate of the cumulative distribution function. This algorithm was shown to outperform EMOS methods for postprocessing of both wind speed and temperature forecasts by Taillardat et al. (2016) and for precipitation by Whan and Schmeits (2018), and will therefore be used as a benchmark in this research.

We use the Python package Scikit-garden to implement quantile regression forests. Within this package there is no option to obtain a full cumulative distribution function. Therefore an alternative prediction function is used which outputs the average of the empirical cumulative distribution functions of the leaves of every tree in the random forest.

For quantile regression forests the most important hyperparameters are the minimum leaf size of the trees and the amount of randomization. We can control the amount of randomization in the random forest by varying the size of the random subset of predictors that is used for splitting at every step. Other hyperparameters that are explored are the choice of the impurity function and the number of trees.

We train quantile regression forests using either the observations or the residuals of linear regression. The second approach could be beneficial for two reasons. First, quantile regression forests cannot extrapolate outside the range of the

TABLE 3. Predictors considered in our hyperparameter search. The predictors that gave the most skillful forecasts in cross validation are indicated in boldface.

No.	Predictor
1	**Sine and cosine of the wind direction at a height of 10 m**
2	**Wind speed at a height of 10 m**
3	**Surface roughness**
4	**Meridional/zonal wind components at 925 hPa**
5	**Mean sea level pressure**
6	Total kinetic energy
7	Humidity at surface level
8	Geopotential height 500 hPa
9	Temperature at surface level
10	Meridional and zonal wind components at 850 hPa
11	Day of the year

training data. As linear regression is able to extrapolate, we may be able to obtain a better model for higher wind speeds by combining QRF and linear regression. Second, random forests split the data into boxes based on which split minimizes the total impurity. If the relationship between the response variable and a single predictor is linear, then it may take random forests many splits to represent this relationship. Splits based on other variables are as a result made with limited information. Fitting to the residuals of a linear model can reduce this effect.

b. Conditional density estimation using neural networks

In this work we consider three different methods for conditional density estimation using convolutional neural networks. These methods are quantized softmax, kernel mixture networks with Gaussian kernels, and parametric density estimation with a truncated normal distribution. The first method adds an additional quantized softmax output layer to the neural network to create an estimate of the conditional probability density function by a histogram with predefined bins. This method is also used for conditional density estimation using fully connected neural networks. The second method fits a mixture of normal distributions, where the mean of every Gaussian is fixed but the weights in the mixture and the standard deviations of the Gaussians are learned by the network. The means of the Gaussians are taken to lie on a regularly spaced grid between -15 and $15\,\mathrm{m\,s^{-1}}$, i.e., the set of means of the residuals is given by $\{-15, -15 + 30/N, \ldots, -15 + 30(N - 1)/N\}$. The number of kernels N is used as a hyperparameter in this study. The third method fits a truncated normal distribution. In this case the network learns the two parameters of the distribution. In case the network is trained directly on observations, the normal distribution is truncated at zero. If the network instead trains on residuals, then the distribution is truncated at minus the forecast of the linear regression in order to ensure that negative wind speeds cannot be predicted. In the appendix a more detailed description of the three methods is given.

We train the neural networks by empirical loss minimization. As potential loss functions, we consider the continuous ranked probability score (CRPS) and the negative log-likelihood. The CRPS of a given conditional cumulative distribution function estimate $\hat{F}$ (associated with a conditional probability density function estimate) and a training datum (x, y) is defined by

$$\mathrm{CRPS}[\hat{F},(x,y)] = \int_{-\infty}^{\infty} [\hat{F}(c|x) - 1_{[y,\infty)}(c)]^2\, dc.$$

The log(arithmic) score is defined as the negative log-likelihood of a given conditional density function estimate $\hat{p}$ and training datum (x, y), and is given by

$$\mathcal{L}[\hat{p},(x,y)] = -\log[\hat{p}(y|x)].$$

c. Fully connected neural networks

In this section we give a concise description of the fully connected neural networks used in this work. For a general introduction to neural networks and the standard terminology used in this work we refer to Goodfellow et al. (2016). We explore networks whose first part is a stack of n blocks, each of which contains a dense layer of size m followed by a ReLU activation function and a dropout layer. We have also explored architectures with batch normalization layers, as are used in the convolutional neural networks (see section 3d), but found that these did not increase the performance.

For the fully connected neural networks we use the quantized softmax method for conditional density estimation. We minimize the empirical loss associated with either the CRPS or the negative log-likelihood using adaptive moment estimation (Adam; Kingma and Ba 2014), a variant of stochastic gradient descent that is very popular in deep learning. We use early stopping to determine the number of epochs (the number of times the training data are used during training).

The neural networks used in this research were programmed using Keras (Chollet et al. 2015), with TensorFlow as backend (Abadi et al. 2015). Adam was employed using default options for all parameters other than the learning rate decay parameter.

As in the case of QRF, the fully connected neural network is trained using either the observations or the residuals of linear regression. For neural networks applying linear regression is hypothesized to give better results due to the fact that lower wind speeds are much more prevalent in the training dataset. Output neurons that are related to high wind speeds therefore need to be activated in only a very small sample of the data. In case of direct training, we use a softmax layer with 30 output bins, where every bin (of size $1\,\mathrm{m\,s^{-1}}$) represents a different wind speed ranging from 0 to $30\,\mathrm{m\,s^{-1}}$. For the neural network that is trained on the residuals of linear regression as target variables, we use 300 identically sized output bins. In this case, every bin represents a different value of the residual ranging between -15 and $15\,\mathrm{m\,s^{-1}}$. In the linear regression case we use a higher resolution as the errors of linear regression can take on any value, whereas the actual measurements are rounded to the nearest full meter per second. When training on the residuals, we add Gaussian noise with mean zero and variance σ^2 to the target variables to smoothen the results. This is necessary as the number of bins is rather large compared to the number of values in the training dataset.

The hyperparameter search is performed on the number of blocks n and layer size m, dropout rate, ℓ_1-regularization strength, learning rate, batch size and, in case we use linear regression, the label noise variance σ^2. Moreover, we investigate the effect of using the CRPS and the negative log-likelihood during training. We furthermore check the same potential predictor variables as for QRF (Table 3). Finally, we used the learning rate decay parameter of Adam as a hyperparameter.

d. Convolutional neural networks

When applying neural networks to high-dimensional input data, such as images, the number of trainable parameters becomes very large. Convolutional neural networks are neural networks that are specialized in analyzing images, by limiting the number of parameters in the network based on the structure of the task at hand. This is achieved using two techniques. The first technique is parameter sharing: a number of parameters are assigned the same value to ensure that the same transformation is applied everywhere in the input image. In object detection tasks, this ensures that it is irrelevant where the object we want to detect is located in the image. The second technique is local connectivity, i.e., only connecting neurons that are related to pixels from an input image, which are close to each other. This is based on the assumption that the relation between pixels that are close to each other is important.

For the same reason as for fully connected neural networks, we investigate training of CNNs on the residuals of linear regression. An additional motivation, which is more specific to CNNs, is that we can use local information for linear regression. The strength of a CNN is based on the translation invariance of the spatial patterns that it needs to learn. The wind speed at a particular weather station is, however, expected to be strongly dependent on the wind speed forecast of the NWP model at that station. The translation invariance of the convolutional layers is therefore not suited for predictions at a specific weather station. Features in the forecast that correlate to the bias and the forecast uncertainty are expected to be less local in nature and should therefore be better suited for analysis using CNNs.

The CNNs all receive two different inputs. The first input is the full spatial forecast of the wind speed for a certain region around the weather station, which provides the corresponding observation. This is the input that is received by the convolutional part of the network. The second input contains the other variables, obtained from the nearest grid boxes around the station, similar to what is used for QRF and the fully connected neural networks.

The final architecture of the network is shown in Fig. 1. The convolutional part of the network consists of n_{conv} layers with m_{conv} filters, where the same number of filters is used in every layer. Each of these convolutional layers is followed by a Relu activation function, a batch normalization layer and a max pooling layer, where we use a step size of 2×2 for the Max Pooling layer and a filter size of 3×3 for the convolutional layer. For the fully connected part of the network every dense layer was followed by a Relu activation function, a batch normalization layer and a dropout layer.

For the convolutional neural network we compare all three methods for conditional density estimation that were discussed

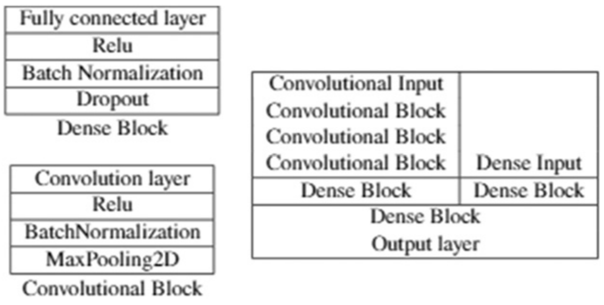

FIG. 1. Final convolutional neural network architectures, where the size of the fully connected layers and convolutional layers are as given in Table 5. The output layer is a fully connected layer where the size and activation functions depend on the method used.

in section 3b, i.e., quantized softmax, kernel mixture networks, and fitting a truncated normal distribution. We again train the networks by minimizing the empirical loss associated with either the CRPS or the negative log-likelihood using Adam.

The size of the output layer of the convolutional neural networks depends on which conditional density estimation method is used. For quantized softmax, from here on referred to as CNN_LR, the output layer has size 300, as is also used for NN_LR. For fitting a truncated normal, from here on referred to as CNN_LR_N0, we ruse two output neurons (corresponding to the two parameters of the distribution) and for the kernel mixture network, from here on referred to as CNN_LR_KMN, we need two output neurons for every kernel that we use.

e. Predictor and hyperparameter selection

Due to the large number of different parameters and potential predictor variables, it is computationally infeasible to select variables and hyperparameters using a full grid search. To select predictors, model architectures, and hyperparameters we use a random search through a large range of options for both the neural networks and the random forests. Based on the results of a large sample of models we narrow down the range for hyperparameters that are most important and repeat the procedure. In this way we can search through a large space of possible models with limited computational resources. The initial range of the hyperparameters is selected large enough to ensure that models with parameter values on the high and low end of the range perform clearly worse than models with intermediate parameter values. In this, we enlarge the chances that good parameter values are contained in the initial range. The only exception to this rule is the selection of the batch size for the CNNs: in this case we checked batch sizes that are powers of 2 and select the largest batch size that is computationally feasible for the computer used.

4. Results

a. Variable selection and hyperparameter tuning

As has been explained above, some models have been trained on the errors of linear regression instead of on the

TABLE 4. Hyperparameters for the selected models.

Hyperparameter	NN	NN_LR
No. of layers	2	3
Layer size	106	106
Batch size	256	256
Learning rate	3.47×10^{-3}	1.57×10^{-3}
Dropout rate	0.030	0.188
Loss function	log-likelihood	log-likelihood
Decay parameter	5.0×10^{6}	8.4×10^{4}
σ^2 noise	0	0.315

measurements directly. In these cases, linear regression was fitted on the mean values of 10 m wind speed, surface roughness and 925 hPa meridional and zonal wind components (predictor variables 2, 3, and 4 as defined in Table 3) on an area of 12.5 km × 12.5 km around the stations. These variables were selected through forward stepwise selection, a greedy algorithm that adds predictors successively based on which predictors reduce the mean squared error (MSE) the most. As the MSE did not improve significantly after these predictors had been selected, all other candidate variables have been left out. However, all candidate predictor variables have been used in the nonlinear methods, as they improved results in all cases.

The best QRF models, as determined by the hyperparameter search, have the following characteristics. The predictor data contain the maximum, minimum and mean value of the predictors that are marked in bold in Table 3. The best results were obtained by using this full set of predictors for splitting at every step, so that decorrelation between the trees only occurs through bootstrapping on the training set. For the impurity function we compared the mean squared error to the mean absolute error, and found the former to give the best results. For the random forest trained on the wind speed measurements, hereafter referred to as QRF, we have used a minimum leaf size of 30. For the random forest trained on the residuals of linear regression (QRF_LR), we have used a minimum leaf size of 42. Oversampling the data, such that training samples corresponding to high wind speed days are shown to the random forest more often during the training phase, was tried, but this appeared to have a negative impact on the results. This may be due to the fact that this leads to a large number of copies of outliers in the training set, which do not generalize well. Furthermore, oversampling based on observations gives a bias for higher wind speed values, and therefore oversampling based on HA40 wind speed forecasts would probably have been a better choice. Less naive oversampling methods with data augmentations might be more useful still, but were not tried. We used 100 trees during the first hyperparameter search and 500 trees for the final model.

The neural network trained on the wind speed measurements themselves, hereafter referred to as NN, appears to give the best results if it uses the maximum and mean value of the sine and cosine of 10 m wind direction, 10 m wind speed, and surface roughness (predictor variables 1, 2, and 3 from Table 3). The neural network trained on the residuals, hereafter referred to as NN_LR, gives the best results when trained on the means of the bold predictor variables shown in Table 3 and the maximum and minimum value of the wind speed. The ℓ_1-regularization did not appear to improve the results and was left out completely for both methods. The values of the other hyperparameters are shown in Table 4.

The convolutional networks have all been trained on the residuals of linear regression. Convolutional neural networks trained on the observations were found to be not skillful in preliminary testing. This was partly due to the fact that networks trained on the observations directly took longer to converge and converged to poor values more often than models trained on the residuals. This resulted in a significantly slower hyperparameter search. No real difference in performance is observed between the CRPS and log-likelihood as loss functions, neither in training time nor in the final result. The log-likelihood is, however, more sensitive to the initialization, since a poor initial estimate leads to exploding gradients. This is less of an issue when using the CRPS, since for a deterministic forecast the CRPS is equal to the mean absolute error, for which the derivative is piecewise constant. This results in a more stable behavior during the training phase.

The hyperparameters used in the hyperparameter search and the selected values of each of these hyperparameters are shown in Table 5.

TABLE 5. Hyperparameters of CNNs.

Hyperparameter	CNN_LR_N0	CNN_LR_KMN	CNN_LR
Input grid size	100 × 100	60 × 60	60 × 60
Variables	1, 2, 3, 4, 5	1, 2, 3, 4, 5	1, 2, 3, 4, 5
Layer size	60	80	80
Size of convolutional layers	16	16	16
Batch size	128	128	128
Learning rate	0.0013	0.000 53	0.000 728 3
Loss function	CRPS	CRPS	Log-likelihood
Dropout rate	0.1028	0.072	0.0888
Decay parameter	2.633×10^{-6}	4.098×10^{-5}	4.10×10^{-7}
Noise	0.315	0.26218	0.322
No. of kernels	—	60	—

TABLE 6. Continuous ranked probability score of different methods in cross validation, with boldface values indicating the best scores. Fold 1, Fold 2, and Fold 3 refer to the verification fold in cross validation (Table 2).

Method	Fold 3	Fold 1	Fold 2
NN	0.824	0.898	0.914
NN_LR	0.828	0.865	0.889
QRF	0.814	0.861	0.888
QRF_LR	0.819	0.871	0.900
CNN_LR_KMN	0.794	0.830	0.861
CNN_LR_N0	0.772	**0.806**	0.848
CNN_LR	**0.769**	0.810	**0.839**

b. *Verification results for models trained on two-thirds of the training data*

The CRPS results for the three different cross-validation folds for the best models of all methods are shown in Table 6. These results show that convolutional neural networks outperform QRF and NN on all threefolds in cross validation. Hyperparameters were selected based on these results, however, and therefore we have also checked the CRPS on the independent test set (Table 7). On the latter set, three different forecasts were verified using every method. Each of these forecasts is based on the model trained on a different training set as used in the cross validation, in order to obtain an estimate of the variation in the results when different training data are used. A downside of this procedure is that it may favor neural networks over quantile regression forests, due to the fact that we use early stopping based on the validation set in training the neural networks. Therefore a final comparison is made between the CNNs and QRF when they are trained on the full training dataset (section 4c).

In Table 7 the results are shown for the root-mean-square error, the mean absolute error, the CRPS, and the log score based on the independent test set. These results show that adding spatial information through convolutions reduces the error of both the deterministic forecast (i.e., the mean and the median of the probabilistic forecast for the RMSE and MAE, respectively) and the probabilistic forecast. Furthermore we can see that applying linear regression improves the scores of the fully connected neural networks. However, it does not improve the scores of QRF, except for the RMSE. In Figs. 2a–c the Brier skill score relative to QRF is shown for the three different training sets. From this figure it is clear that convolutional neural networks are more skillful than the other methods at higher wind speeds. For wind speeds above 18 m s^{-1} their performance becomes worse again; however, in this range there is not enough data to draw any conclusions. Figure 2 also shows that learning the residuals of linear regression mainly helps to improve forecasts for higher wind speeds, while for low wind speeds the results become worse for both neural networks and random forests.

Figure 2d shows the probability integral transform (PIT) diagram of all the methods. In this figure we can see a clear difference between the models trained on the wind speed observations and models trained on the residuals of linear regression for QRF and the fully connected neural networks. The methods trained on the residuals lie closer to the diagonal, implying that on average they make a better estimate of the probabilities. It is surprising, however, that this does not hold for the CNNs that are trained on residuals. For QRF and the CNNs we see that the PIT curve lies under the diagonal, which means that for these methods observations fall in the higher quantiles of the estimated distribution more often than expected. This implies that the probability of higher wind speeds is underestimated by these methods.

c. *Verification results for models trained on the full training dataset*

We make a final comparison of QRF and the CNNs by training the models on the entire training dataset; fully connected neural networks are omitted since they showed poorer performance in the results of the previous subsection. For the convolutional neural networks the number of epochs was chosen to be 2/3 of the average number of epochs that gave the best results in cross validation, i.e., 6, 12, and 16 for CNN_LR_N0, CNN_LR_KMN and CNN_LR, respectively. This choice ensures that the number of training steps is the same as in section 4b. The results obtained for the CNNs are comparable, in terms of CRPS (Table 8), to the case where they were trained on only part of the training data (Table 7). We see in Table 8 that the CNNs still outperform QRF. The BSS of the CNNs, compared to QRF, is slightly lower, however, for wind speeds between 12 and 16 m s^{-1} (cf. Figs. 3 and 2a–c).

Figure 3 shows the Brier skill scores of the models trained on the full dataset with respect to both the station climatology (left panel) and QRF (right panel). In this figure we include a

TABLE 7. Results on the independent test set with boldface values indicating the best scores. Fold1, Fold2, and Fold3 refer to the fold that is left out of the training data (Table 2). The standard deviation in the CRPS was estimated by block bootstrapping 1000 times.

	RMSE			MAE			CRPS			Log score		
	Fold3	Fold1	Fold2	Fold3	Fold1	Fold2	Fold3	Fold1	Fold2	Fold3	Fold1	Fold2
NN	2.457	2.331	2.391	1.144	1.117	1.124	0.820 ± 0.012	0.799 ± 0.011	0.809 ± 0.011	4.42	4.36	4.37
NN_LR	2.204	2.126	2.176	1.113	1.089	1.100	0.793 ± 0.012	0.779 ± 0.011	0.786 ± 0.012	3.99	3.98	3.98
QRF	2.244	2.220	2.245	1.080	1.076	1.085	0.782 ± 0.012	0.776 ± 0.012	0.779 ± 0.012	4.03	4.02	4.02
QRF_LR	2.157	2.151	2.154	1.089	1.090	1.087	0.780 ± 0.012	0.781 ± 0.012	0.780 ± 0.012	4.05	4.04	4.04
CNN_LR_KMN	1.968	1.886	1.922	1.056	1.046	1.053	0.752 ± 0.011	0.744 ± 0.011	0.748 ± 0.011	3.96	3.97	3.95
CNN_LR_N0	**1.818**	1.861	2.117	**1.012**	1.029	1.088	**0.722** ± 0.011	0.732 ± 0.011	0.770 ± 0.013	**3.90**	3.92	3.96
CNN_LR	1.851	**1.814**	**1.889**	1.014	**1.004**	**1.032**	0.724 ± 0.011	**0.718** ± 0.011	**0.733** ± 0.011	3.91	**3.91**	**3.92**

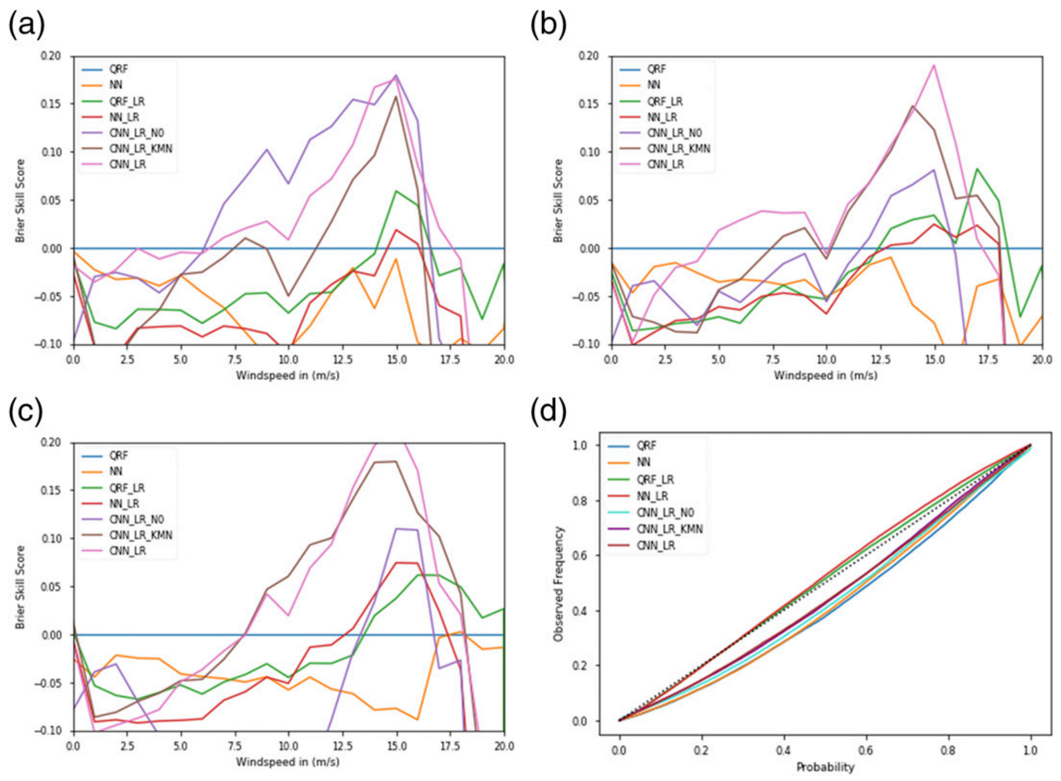

FIG. 2. (a)–(c) Brier skill scores of the different methods relative to QRF, for predictions trained on three different training sets (see Table 2). (d) PIT diagram for the forecasts of the different methods for the three cross-validation sets combined.

bootstrap estimate of the standard deviation, obtained by block bootstrapping 1000 times, i.e., by drawing data from all stations of a single date at once because of spatial correlation. From this figure it is clear that the CNNs perform better than QRF for higher wind speeds (~11–15 $\mathrm{m\,s^{-1}}$). Furthermore, it is clear that for wind speeds above 15 $\mathrm{m\,s^{-1}}$ the uncertainty in the Brier skill scores is much larger than the difference in Brier skill scores between the methods.

Figure 4 shows reliability diagrams for thresholds of 5, 10, and 15 $\mathrm{m\,s^{-1}}$. Here we can see that for 5 $\mathrm{m\,s^{-1}}$ the forecasts of QRF are better calibrated, but both the CNNs and QRF_LR forecasts are somewhat sharper. For 10 $\mathrm{m\,s^{-1}}$ the QRF forecasts are still better calibrated, but they give less often a high probability of exceeding this threshold and are slightly less sharp. Finally, for 15 $\mathrm{m\,s^{-1}}$ we can see that QRF is significantly worse at predicting these events when they are likely; both the calibration and sharpness are worse than for CNNs.

d. Geographic differences in CRPSS

Figure 5 shows the difference in continuous ranked probability skill score (CRPSS), with respect to the climatology, between QRF and the CNNs for the different stations. This shows that CNN is the most skillful method for almost all stations, although the CRPSS difference between the methods is relatively small for most stations.

By visualizing the activations of the convolutional layers for the CNNs one can observe that this method is able to detect the Dutch coastline [see Veldkamp (2020) for details]. Based on this observation, we could hypothesize that the CNNs are more skillful for higher wind speeds due to a higher skill for coastal stations. Figure 6 shows the CRPSS of CNN_LR with respect to QRF on a map of the Netherlands. We cannot see a clear indication of higher CRPSS values of CNN_LR for coastal stations, so that the higher skill of the CNNs is not fully explained by its ability to differentiate between coastal and noncoastal stations.

5. Conclusions and discussion

We have shown that for +48 h wind speed forecasts convolutional neural networks can be of added value for statistical

TABLE 8. The root-mean-square error, mean absolute error, continuous ranked probability score, and log score of different methods for the independent test set, trained on the full training dataset (Table 2) with boldface values indicating the best scores.

Method	RMSE	MAE	CRPS	Log score
Climatology	2.676	2.033	1.445	∞
Linear Regression	2.399	1.170	—	—
QRF	2.217	1.077	0.776	4.02
QRF_LR	2.124	1.081	0.774	4.03
CNN_LR_KMN	1.905	1.034	0.740	3.96
CNN_LR_N0	1.891	1.037	0.735	**3.91**
CNN_LR	**1.889**	**1.033**	**0.731**	3.93

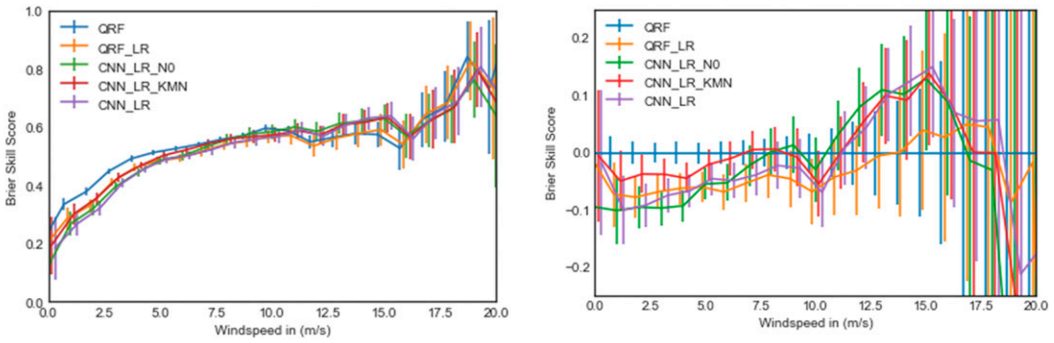

FIG. 3. Brier skill scores relative to the (left) station climatology and (right) QRF, for models trained on the full training dataset. The error bars represent the estimates of the standard deviation obtained by block bootstrapping the test data 1000 times. Block bootstrapping was used to ensure that spatial correlation between stations is accounted for.

postprocessing. Convolutional neural networks outperform quantile regression forests and fully connected neural networks, in terms of CRPS, in all the three cross-validation sets, and in terms of CRPS, log score, MAE and RMSE in the final independent test set. Besides, we have compared the CNNs using three different density estimation methods [quantized softmax (QS), kernel mixture networks, and fitting a truncated normal distribution], and found the probabilistic forecasts based on the QS method to be best.

The Brier skill score shows that CNNs outperform QRF for higher wind speeds that are more important in weather forecasting because of their potential impact on society. In contrast, for wind speeds up to ~10 m s^{-1} QRF has both a better Brier skill score and is better calibrated. The poor performance of the CNNs with respect to QRF in the lower wind speed range could be explained as an effect of using ordinary least squares regression. The latter assumes errors that are symmetrically distributed around zero and therefore does not perform well for low wind speeds, as this method implicitly assumes that negative wind speeds are possible. This could be mitigated by performing a variant of ordinary least squares that excludes this possibility. For wind speeds above 15 m s^{-1} the uncertainty in the Brier skill score grows very fast and conclusions for this range can therefore not be drawn. This is mainly caused by a lack of cases with high wind speeds in the available dataset. The test set, for example, only includes a single measurement above 19 m s^{-1}. An obvious solution for this would be to obtain more data by obtaining reforecast data for more years, assuming these years contain more climatologically extreme wind speeds. A less costly solution to this

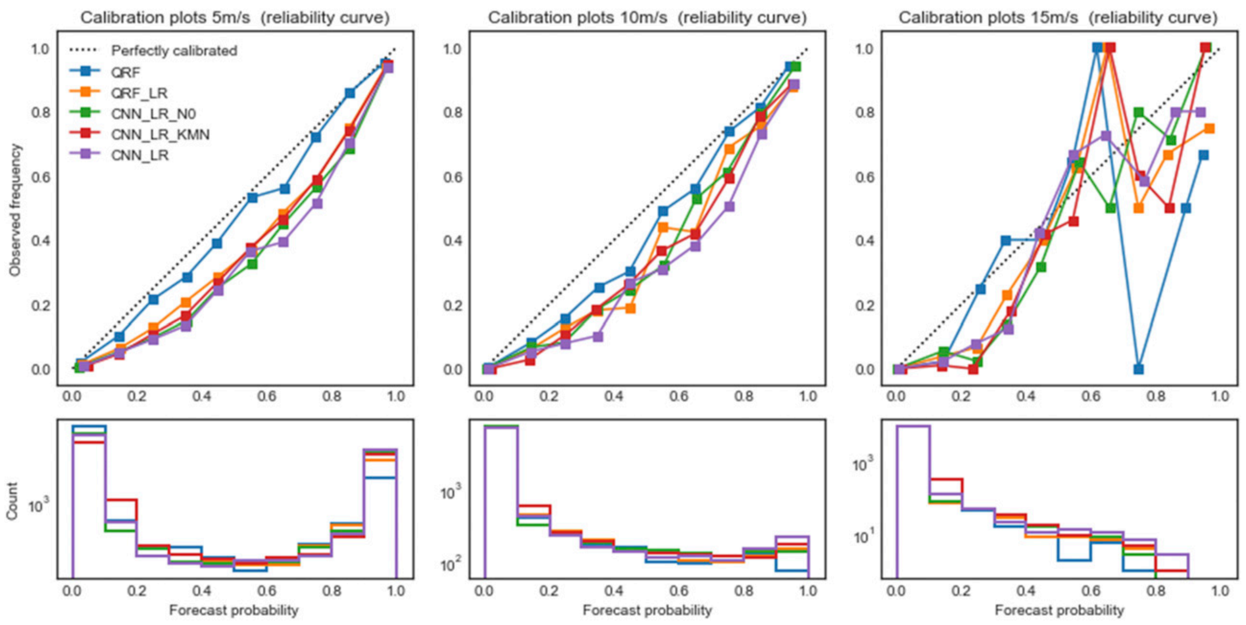

FIG. 4. Reliability diagram for the CNNs and QRF, trained on the full training dataset, for thresholds of (left) 5, (center) 10, and (right) 15 m s^{-1}. The exceedance frequencies for these thresholds in our test dataset are 49.2%, 9.3%, and 1.0%, respectively.

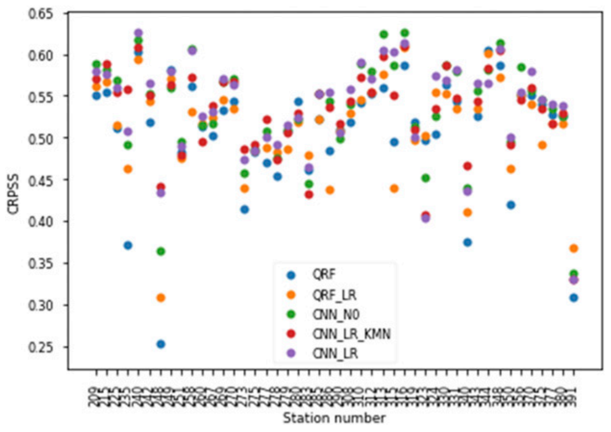

FIG. 5. CRPSS with respect to station climatology of different methods based on the models trained on the full training dataset. Station numbers are explained in Table 1.

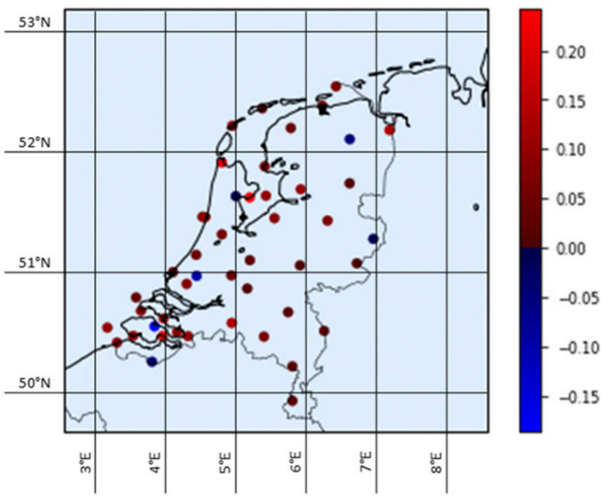

FIG. 6. CRPSS of CNN_LR with respect to QRF. Here positive values imply that CNN_LR is more skillful than QRF.

problem could be to reforecast days in the past with more extreme weather, such as days on which weather warnings were issued, instead of reforecasting full years only, as is currently done.

Although convolutional neural networks proved to be most skillful in our study, a drawback of this method is that it is difficult to interpret for a meteorologist. In the appendix of Veldkamp (2020) one can find a few figures showing the activations in the convolutional layers of the network for a number of days. These do not give a clear indication of which input features are important, although the coast line can be clearly seen. In future research it would be a good addition to use explainability methods, such as layer-wise relevance propagation (Bach et al. 2015), to visualize which parts of an input image are most relevant for the prediction made by a convolutional neural network. This could be especially useful when fitting a truncated normal distribution, as in this case it may be possible to distinguish between features that are relevant in correcting the bias and features that are relevant in predicting the spread. In an ideal case, identifying features that are able to correct a large bias or reduce the spread might even help in identifying shortcomings in the NWP model.

Another drawback of convolutional neural networks is the fact that they require a large amount of training data and are therefore probably less suited for local (i.e., for each station separately) postprocessing. In this study we focused on training global models (i.e., for all stations at once) without using station specific information apart from model surface roughness. This has the benefit that there is more training data available and that the resulting models can be used to postprocess the weather forecast for all grid cells instead of only for specific stations. For countries with mountainous areas topographical predictors should be added in such a global model framework (e.g., Rasp and Lerch 2018).

At the time this study was conducted not enough data was available from the KNMI HARMONIE-AROME ensemble forecasts. As many current statistical postprocessing studies are based on ensemble output, an important next step would therefore be to investigate if convolutional neural networks also add skill when potential predictors are taken from ensemble forecasts.

Acknowledgments. We thank the following people from KNMI: Toon Moene for executing the reforecasting runs of the HARMONIE-AROME model, and Andrea Pagani and Dirk Wolters for assisting with the practical implementation of the deep learning methods used in this paper. Besides, we are grateful to the 3 anonymous reviewers for their comments, which have helped to improve a previous version of the manuscript. S.D. acknowledges funding by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under SPP 1798 (COSIP - Compressed Sensing in Information Processing).

APPENDIX

Conditional Density Estimation

In this appendix we briefly discuss the three methods that have been used to estimate the conditional probability density function of the neural networks.

a. Quantized softmax

Quantized softmax (Oord et al. 2016) is a simple method to obtain an estimate for the conditional density using neural networks. The goal of the method is to approximate the conditional density by a histogram with m predetermined bins $A_1, \ldots, A_m$. For this purpose we construct a neural network (with a linear output layer) with m output neurons and apply the softmax function:

$$\text{Softmax}(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{m} e^{z_j}}, \quad i = 1, \ldots, m$$

to the output of the last layer. We can turn the resulting output $w(x)$ (associated with an input datum x) into an estimate $\hat{p}(y|x)$ for the conditional probability density function by setting:

$$\hat{p}(y|x) = \sum_{i=1}^{m} \frac{1}{\mathrm{Vol}(A_i)} 1_{A_i}(y) w(x)_i ,$$

where $\mathrm{Vol}(A_i)$ is the size of bin A_i. The set of probability densities that can be approximated well by this procedure is controlled by the choice of the bins A_i.

b. *Kernel mixture networks*

The second method used in this paper for conditional density estimation is the kernel mixture network (KMN; Ambrogioni et al. 2017). We describe here directly the variant of KMN with Gaussian kernels, which is used in our study. This variant is very similar to a mixture density network (Bishop 1994). The kernel mixture network estimates the conditional density by a mixture of Gaussians in which the means are fixed and the weights and variances are learned by a neural network. Let $Y = \{y_1, \ldots, y_m\}$ be a subset of the label space containing the kernel centers. Let $\phi(y) = (1/\sqrt{2\pi})e^{-y^2/2}$ denote the standard Gaussian density. We construct a neural network (with a linear output layer) with $2m$ output neurons. To the first m outputs we apply the softmax function and denote the resulting output for a given input datum x by $w(x)$. The entries of $w(x)$ are the weights in the mixture of Gaussians. To each of the last m outputs of the network we apply the softplus function:

$$\mathrm{Softplus}(t) = \log(1 + e^t)$$

and let $\sigma(x)$ denote resulting output. The entries of $\sigma(x)$ are the standard deviations in the mixture of Gaussians. The softplus function ensures that the entries of $\sigma(x)$ are positive and prevents them from becoming too small, which could cause numerical instability. Together, $w(x)$ and $\sigma(x)$ yield an estimate of the conditional probability density function given by

$$\hat{p}(y|x) = \sum_{i=1}^{m} \frac{w(x)_i}{\sigma(x)_i} \phi\left[\frac{y - y_i}{\sigma(x)_i}\right].$$

In the above we could also learn the centers of the network: this procedure is exactly a mixture density network Bishop (1994), which has not been tested in this study. In Ambrogioni et al. (2017), the negative log-likelihood is used for training the network. It is also possible to use the CRPS for training, as a closed form expression is known for the CRPS of a mixture of Gaussians (Grimit et al. 2006): if F is the cumulative distribution function of a mixture of m Gaussians with weights $w_1, \ldots, w_m$, means $\mu_1, \ldots, \mu_m$, and variances $\sigma_1^2, \ldots, \sigma_m^2$, then

$$\mathrm{CRPS}(F, y) = \sum_{i=1}^{m} w_i A(y - \mu_i, \sigma_i^2) - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} w_i w_j A(\mu_i - \mu_j, \sigma_i^2 + \sigma_j^2),$$

where

$$A(\mu, \sigma^2) = \mu\left[2\Phi\left(\frac{\mu}{\sigma}\right) - 1\right] + 2\sigma\phi\left(\frac{\mu}{\sigma}\right)$$

and Φ is the cumulative distribution function of a standard Gaussian. To our knowledge the CRPS has not been used before for kernel mixture networks or quantized softmax. It has, however, been applied with success to train mixture density networks in D'Isanto and Polsterer (2018) and Rasp and Lerch (2018).

c. *Fitting a truncated normal distribution*

The third method considered in this study uses a neural network to learn the parameters of the normal distribution that has been truncated at zero. As was discussed in the introduction, this distribution has been successfully used for post-processing of wind speed forecasts. As the basis for the method we construct a neural network with two output neurons. Let $\mu(x)$ denote the first output and let $\sigma(x)$ be the result of applying the softplus function to the second output. The corresponding estimate of the conditional probability density function is then given by

$$\hat{p}(y|x) = \frac{\frac{1}{\sigma(x)}\phi\left[\frac{y - \mu(x)}{\sigma(x)}\right]}{1 - \Phi\left[-\frac{\mu(x)}{\sigma(x)}\right]}, \quad \text{if} \quad y > 0,$$

and $\hat{p}(y|x) = 0$ else. We can again use the log-likelihood and CRPS for training. If F denotes the cumulative distribution function of a normal distribution truncated at zero, then the CRPS is given by

$$\mathrm{CRPS}(F, y) = \frac{\sigma}{p^2}\left[sp(2\Phi(s) + p - 2) + 2p\phi(s) - \frac{1}{\sqrt{\pi}}\Phi\left(\frac{\mu\sqrt{2}}{\sigma}\right)\right],$$

where $p = \Phi(\mu/\sigma)$ and $s = (y - \mu)/\sigma$ (Thorarinsdottir and Gneiting 2010).

REFERENCES

Abadi, M., and Coauthors, 2015: TensorFlow: Large-scale machine learning on heterogeneous systems. https://arxiv.org/abs/1603.04467.

Ambrogioni, L., U. Güçlü, M. A. van Gerven, and E. Maris, 2017: The kernel mixture network: A nonparametric method for conditional density estimation of continuous random variables. https://arxiv.org/abs/1705.07111.

Bach, S., A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek, 2015: On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLOS ONE*, **10**, e0130140, https://doi.org/10.1371/journal.pone.0130140.

Bakker, K., K. Whan, W. Knap, and M. Schmeits, 2019: Comparison of statistical post-processing methods for probabilistic NWP forecasts of solar radiation. *Sol. Energy*, **191**, 138–150, https://doi.org/10.1016/j.solener.2019.08.044.

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

Bengtsson, L., and Coauthors, 2017: The HARMONIE–AROME model configuration in the ALADIN–HIRLAM NWP system. *Mon. Wea. Rev.*, **145**, 1919–1935, https://doi.org/10.1175/MWR-D-16-0417.1.

Bishop, C. M., 1994: Mixture density networks. Tech. Rep. NCRG/94/004, Aston University, Birmingham, AL, 24 pp.

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

Chollet, F., and Coauthors, 2015: Keras. Accessed 10 May 2019, https://keras.io.

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.

Dupuy, F., O. Mestre, and L. Pfitzner, 2020: Arpege cloud cover forecast post-processing with convolutional neural network. *EGU General Assembly 2020*, EGU2020-18325, https://doi.org/10.5194/egusphere-egu2020-18325.

Gagne, D. J., II, S. E. Haupt, D. W. Nychka, and G. Thompson, 2019: Interpretable deep learning for spatial analysis of severe hailstorms. *Mon. Wea. Rev.*, **147**, 2827–2845, https://doi.org/10.1175/MWR-D-18-0316.1.

Glahn, H. R., and D. A. Lowry, 1972: The use of model output statistics (MOS) in objective weather forecasting. *J. Appl. Meteor.*, **11**, 1203–1211, https://doi.org/10.1175/1520-0450(1972)011<1203:TUOMOS>2.0.CO;2.

Gneiting, T., and A. E. Raftery, 2005: Weather forecasting with ensemble methods. *Science*, **310**, 248–249, https://doi.org/10.1126/science.1115255.

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

Goodfellow, I., Y. Bengio, and A. Courville, 2016: *Deep Learning*. MIT Press, http://www.deeplearningbook.org.

Grimit, E. P., T. Gneiting, V. J. Berrocal, and N. A. Johnson, 2006: The continuous ranked probability score for circular variables and its application to mesoscale forecast ensemble verification. *Quart. J. Roy. Meteor. Soc.*, **132**, 2925–2942, https://doi.org/10.1256/qj.05.235.

Hersbach, H., 2000: Decomposition of the continuous ranked probability score for ensemble prediction systems. *Wea. Forecasting*, **15**, 559–570, https://doi.org/10.1175/1520-0434(2000)015<0559:DOTCRP>2.0.CO;2.

Ioannidis, E., K. Whan, and M. Schmeits, 2018: Probabilistic wind speed forecasting using parametric and non-parametric statistical post-processing methods. KNMI Internal Rep. IR-2018-07, 78 pp., http://bibliotheek.knmi.nl/knmipubIR/IR2018-07.pdf.

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

Krizhevsky, A., I. Sutskever, and G. E. Hinton, 2012: ImageNet classification with deep convolutional neural networks. *Adv. Neural Info. Process. Syst.*, **25**, 1097–1105, https://doi.org/10.1145/3065386.

Lagerquist, R., A. McGovern, and D. J. Gagne II, 2019: Deep learning for spatially explicit prediction of synoptic-scale fronts. *Wea. Forecasting*, **34**, 1137–1160, https://doi.org/10.1175/WAF-D-18-0183.1.

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.

Liu, Y., and Coauthors, 2016: Application of deep convolutional neural networks for detecting extreme weather in climate datasets. https://arxiv.org/abs/1605.01156.

Lorenz, E. N., 1963: Deterministic nonperiodic flow. *J. Atmos. Sci.*, **20**, 130–141, https://doi.org/10.1175/1520-0469(1963)020<0130:DNF>2.0.CO;2.

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.

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

Oord, A. V., N. Kalchbrenner, and K. Kavukcuoglu, 2016: Pixel recurrent neural networks. *Proc. 33rd Int. Conf. on Machine Learning*, M. F. Balcan, and K. Q. Weinberger, Eds., Proceedings of Machine Learning Research, Vol. 48, 1747–1756, http://proceedings.mlr.press/v48/oord16.html.

Rasp, S., and S. Lerch, 2018: Neural networks for postprocessing ensemble weather forecasts. *Mon. Wea. Rev.*, **146**, 3885–3900, https://doi.org/10.1175/MWR-D-18-0187.1.

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

Scher, S., and G. Messori, 2018: Predicting weather forecast uncertainty with machine learning. *Quart. J. Roy. Meteor. Soc.*, **144**, 2830–2841, https://doi.org/10.1002/qj.3410.

Scheuerer, M., 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.

——, M. B. Switanek, R. P. Worsnop, and T. M. Hamill, 2020: Using artificial neural networks for generating probabilistic subseasonal precipitation forecasts over California. *Mon. Wea. Rev.*, **148**, 3489–3506, https://doi.org/10.1175/MWR-D-20-0096.1.

Shi, X., Z. Gao, L. Lausen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-C. Woo, 2017: Deep learning for precipitation nowcasting: A benchmark and a new model. https://arxiv.org/abs/1706.03458.

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.

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.

van der Plas, E., M. Schmeits, N. Hooijman, and K. Kok, 2017: A comparative verification of high-resolution precipitation forecasts using model output statistics. *Mon. Wea. Rev.*, **145**, 4037–4054, https://doi.org/10.1175/MWR-D-16-0256.1.

Veldkamp, S., 2020: Statistical postprocessing of windspeed forecasts using convolutional neural networks. M.S. thesis, Universiteit Utrecht, 54 pp., https://dspace.library.uu.nl/handle/1874/393399.

Weyn, J. A., D. R. Durran, and R. Caruana, 2019: Can machines learn to predict weather? Using deep learning to predict gridded 500-hPa geopotential height from historical weather data. *J. Adv. Model. Earth Syst.*, **11**, 2680–2693, https://doi.org/10.1029/2019MS001705.

Whan, K., and M. Schmeits, 2018: Comparing area probability forecasts of (extreme) local precipitation using parametric and machine learning statistical postprocessing methods. *Mon. Wea. Rev.*, **146**, 3651–3673, https://doi.org/10.1175/MWR-D-17-0290.1.

Wilczak, J., and Coauthors, 2015: The Wind Forecast Improvement Project (WFIP): A public–private partnership addressing wind energy forecast needs. *Bull. Amer. Meteor. Soc.*, **96**, 1699–1718, https://doi.org/10.1175/BAMS-D-14-00107.1.