Universiteit Utrecht

Koninklijk Nederlands Meteorologisch Instituut
Ministerie van Infrastructuur en Waterstaat

Faculteit Bètawetenschappen

Multivariate Postprocessing of Temporal Dependencies with Autoregressive and LSTM Neural Networks

Master Thesis
Mathematical Sciences

Daniel Teixeira Soares Tolomei

Supervisors:
Dr. Sjoerd Dirksen
Utrecht University
Dr. Kirien Whan
KNMI
Dr. Maurice Schmeits
KNMI

Second reader:
Dr. Palina Salanevich
Utrecht University

March 30, 2022

Abstract

Weather forecasts issued by Numerical Weather Prediction (NWP) systems often display systematic bias and do not quantify the inherent uncertainty of the forecast. It is the task of statistical postprocessing to use these NWP predictions to issue probabilistic forecasts that address these issues. In this work we focus on multivariate postprocessing, which also requires statistical modelling of the spatial, temporal or inter-variable dependencies. More especifically, we use NWP forecasts from the Harmonie-Arome model to issue multivariate probabilistic forecasts for hourly wind speed predictions from initialization at 0 UTC up to 48h ahead. We propose a new statistical model for multivariate forecasting, the ARMOS(p) model, that exploits the autoregressive property of forecast errors to estimate an explicit parametric distribution, and compare it to a benchmark obtained from a combination of Ensemble Output Statistics (EMOS) with the Schaake Shuffle. We further extend these models by estimating the distribution parameters using neural networks, which incorporate spatial and temporal information from the NWP forecasts by using LSTM and Convolutional layers. In our experiments we verify model performance by computing proper multivariate scores and by performing marginal verification on the test set. The results show that the LSTM/EMOSnet and the ARMOS(2)net improve on the benchmarks, and are the best models overall.

Acknowledgements

Foremost I would like to thank Dr. Sjoerd Dirksen, Dr. Kirien Whan and Dr. Maurice Schmeits for the aid and support they provided during this project. They provided a very positive work environment that helped me keep motivated and challenged, as well as giving me key insights that were essential to the development of the thesis. I'd also like to thank my family and friends for all their support during the challenging times of the pandemic.

Contents

1 Introduction

Weather forecasting is important for activities in various areas of society, from agriculture, to aviation and renewable energy. The Netherlands has been historically heavily impacted by adverse weather conditions, making the task of issuing reliable extreme weather alerts crucial for national security. The particular problem of wind speed forecasting, and more specifically forecasts of extreme winds, is of importance for security of individuals and preservation of property due to recurrent wind storms in north-western Europe, and this will be the focus of this project.

One of the central tools in weather forecasting are Numerical Weather Prediction (NWP) models such as the Harmonie-Arome [5] model of the Royal Netherlands Meteorological Institute (KNMI). These models work via numerical integration over a discretized grid of a system of differential equations that describe the atmospheric flow. The computations are based on physical models and hence produce deterministic outputs. However, given that the atmosphere is a chaotic system [27] it is strictly necessary to quantify uncertainty in forecasts. A common approach is to introduce uncertainty by producing multiple perturbed samples of initial conditions, then use the NWP model to evolve those and produce an ensemble of forecasts from which it is possible to compute statistics. This kind of numerical forecast is called *ensemble* forecast, as opposed to a *deterministic* forecast for which only a single numerical forecast is available. However, these ensembles are computationally expensive to calculate and suffer from systematic bias and underdispersion [13] due to imperfect knowledge of the initial conditions and small-scale physical processes such as cloud formation.

The core motivation of statistical post-processing is to apply statistical methods on the output of NWP models in order to produce corrected (probabilistic) forecasts. One of the most popular post-processing frameworks is Ensemble Model Output Statistics (EMOS) [15], which uses linear regression over an NWP ensemble to fit the parameters of a forecast distribution for a fixed lead time (e.g. 48h ahead). Common distributions include the Truncated Normal or the Log Normal [3, 26]. Althought estimation of parameters via linear regression is most common, it is also possible to use non-linear models such as neural networks instead as has been done in [41].

A central problem in postprocessing is to issue *multivariate probabilistic forecasts*, in which the joint distribution of multiple variables/dimensions must be modelled in order to take into account the dependency structure between them. Examples include spatial, temporal or inter-variable dependencies. A standard procedure is to take a two step approach, first generating marginal probabilistic forecasts independently using a univariate model, and then applying a copula method [36] to estimate the dependencies, outputting a true multivariate distribution.

We focus on postprocessing wind speed forecasts for consecutive lead times (0 - 48h ahead) from the deterministic Harmonie-Arome NWP model. In this case we estimate a joint distribution for all lead times simultaneously from the NWP predictions. The copula approach can be applied as follows: first applying (a variant) of the EMOS model to each lead time to obtain marginal distributions and then reconstructing the temporal dependencies using an empirical copula method like the Schaake Shuffle (SS) [10]. Our aim in this study is to improve on this standard approach by exploring two main alternatives, which are non-exclusive: the first is to model the temporal dependencies *explicitly* via a new statistical model, and the other is to estimate the distribution parameters via neural networks.

This new statistical model, which we call ARMOS(p), exploits the autoregressive dependencies in the forecast errors of the NWP wind speed forecasts to deduce an explicit multivariate model. It can be considered as a multivariate extension of EMOS and it does not require further application of a copula method. The ARMOS(p) model can be directly compared to EMOS/SS, and both can be further extended by using a neural network to estimate the parameters. The ARMOS(p) model builds upon the research in [30, 31], in which an autoregressive extension of EMOS was proposed for the postprocessing of temperature forecasts. The main distinction between our approach and the previously developed one is that in their previous works the AR modification has been used to build corrected NWP forecasts from which the EMOS marginal forecasts are estimated, while our model focuses on developing an explicit multivariate forecast distribution for the weather quantity itself.

Artificial neural networks have been successfully applied to postprocessing in similar contexts. For example, in [41] it was shown that convolutional neural networks can beat state-of-the-art models by exploiting spatial information in deterministic NWP wind speed forecasts. Recurrent architectures have been explored in [2, 38], where the Long-Short Term architecture was used in other postprocessing tasks. Following this line of research, we explore how a combination of convolutional and LSTM layers can improve the forecast of temporally dependent variables such as the sequence of parameters of the marginal forecast distributions.

The experiments in this study provide a wide comparison between the EMOS/SS and ARMOS(p) statistical models with parameter estimation via linear and neural network models. Since the neural networks are way more complex than the simple regression models it was necessary to make the distinction between *global* and *local* linear EMOS and ARMOS. For global models a single regression model is fit for all stations without distinction, while for local models a regression is fitted for each station individually. The local approach is more common in practice and it gives the linear regression the flexibility of station dependent forecasting, providing a better benchmark to the more complicated neural networks, which are always fitted globally. All models are estimated via the Logarithmic Score (LS), and the Energy Score (ES) and Variogram Score (VS) provide alternative multivariate validation metrics [6]. Marginal validation is also performed via analysis of PIT diagrams [12] together with the Brier Skill Score (BSS) and Reliability diagrams [42] at various thresholds, with special attention given to the forecasts of extreme events.

This thesis is structured as follows. In Section 2 we describe the available Harmonie (re)forecast data, its statistical properties and how we performed the validation/training split. Section 3 gives the necessary information regarding multivariate statistical postprocessing, it gives a formal description of the statistical models and of the validation metrics. Section 4 provides an overview of the neural networks used in the experiments. In Section 5 we describe the experiments and results in detail. Section 6 outlines the conclusion and discussion of the results.

2 Data

2.1 Description

The NWP data comes from the deterministic (single run) Harmonie-Arome cycle 40 (HA40) weather forecasts over a 300×300 discretized grid with 2.5km $\times$ 2.5km spacing covering the Netherlands. It consists of hourly forecasts for the variables included in Table 1 given below, ranging from initialization at time 0000 UTC up to the 48h lead time. (Re)Forecast data is available for years from 2015 to 2017 and for the winter of 2018-2019, and we selected days from the extended winter period (October - March).

Predictors
Wind speed at a height of 10m
Mean sea level pressure
Turbulent kinetic energy
Humidity at 2m
Geopotential height at 500 hPa
Wind speed at 850 hPa
Wind speed at 925 hPa

Table 1: HA40 variables.

The predictand data are the wind speed observations at 10m height at 46 Dutch weather stations, shown in Figure 1 and in Table 5, during the same period.

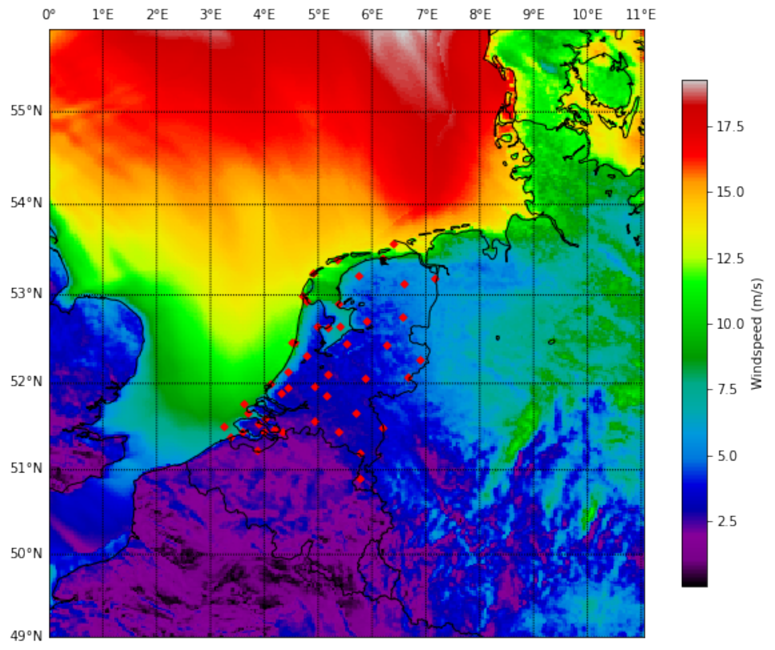


Figure 1: HA40 wind speed forecast for 2015/01/01 at the 48h lead time. Red markers indicate the location of stations from Table 5 of the Appendix.

The predictor data consists of two parts which are extracted from the HA40 predictions for each lead time. The first part contains a grid of hourly HA40 wind speed forecasts at 10m height in a neighbourhood of the station. The station is located at the central grid point and we only consider square neighbourhoods with odd side length (e.g. 25×25 grid points with the station in the center). The second part is a subset of the predictors from Table 1 at the closest grid point to the station. The size of the neighbourhood and the predictor selection can vary and should be determined during model selection. Hence, the input data has two components with dimensions $(49, N, N)$ and $(49, L)$, where N is the grid size, L is the number of selected predictors and 49 is the number of lead times (includes 0h).

The data from the winter of 2018-2019 is separated for independent testing while the data from the years of 2015 - 2017 are used for model training and selection. Furthermore, we split the data from 2015 - 2017 into three groups, the winters of 2015-2016, 2016-2017 and we join the first trimester of 2015 with the last trimester of 2017 to generate the third group. From these 3 groups we select one for validation by analysing the basic statistics displayed in Figure 2, leaving the other two for model training.

Since the stations 229, 285 and 323 (Texelhors, Huibertgat and Wilhelminadorp) have a significant amount of missing samples we remove them completely from all the experiments. Visual inspection indicates that the winter of 2016-2017 was particularly calm, with lower average wind speeds and lower standard deviation, hence validating on it is undesirable because it is too distinct from the other two groups. We also notice that stations 209, 313 and 315 (IJmond, Vlakte v.d. Raan and Hansweert) have lots of missing samples in the winter of 2015-2016. Choosing 2015.0 - 2017.1 for validation would introduce an imbalance, with few training samples for these stations and a higher amount for validation. Therefore we use the winter of 2015-2016 for validation, and the rest of the data is used for training.

2.2 Analysis

We refer to [7] for general concepts in Time Series analysis.

We perform exploratory data analysis in order to extract relevant properties of the training/validation dataset that can guide the statistical modelling at a later stage. Figure 3 shows marginal statistics of the forecast errors of NWP forecasts (difference between forecast and observation) for each lead time and on a few reference stations as well as the global aggregate over all stations.

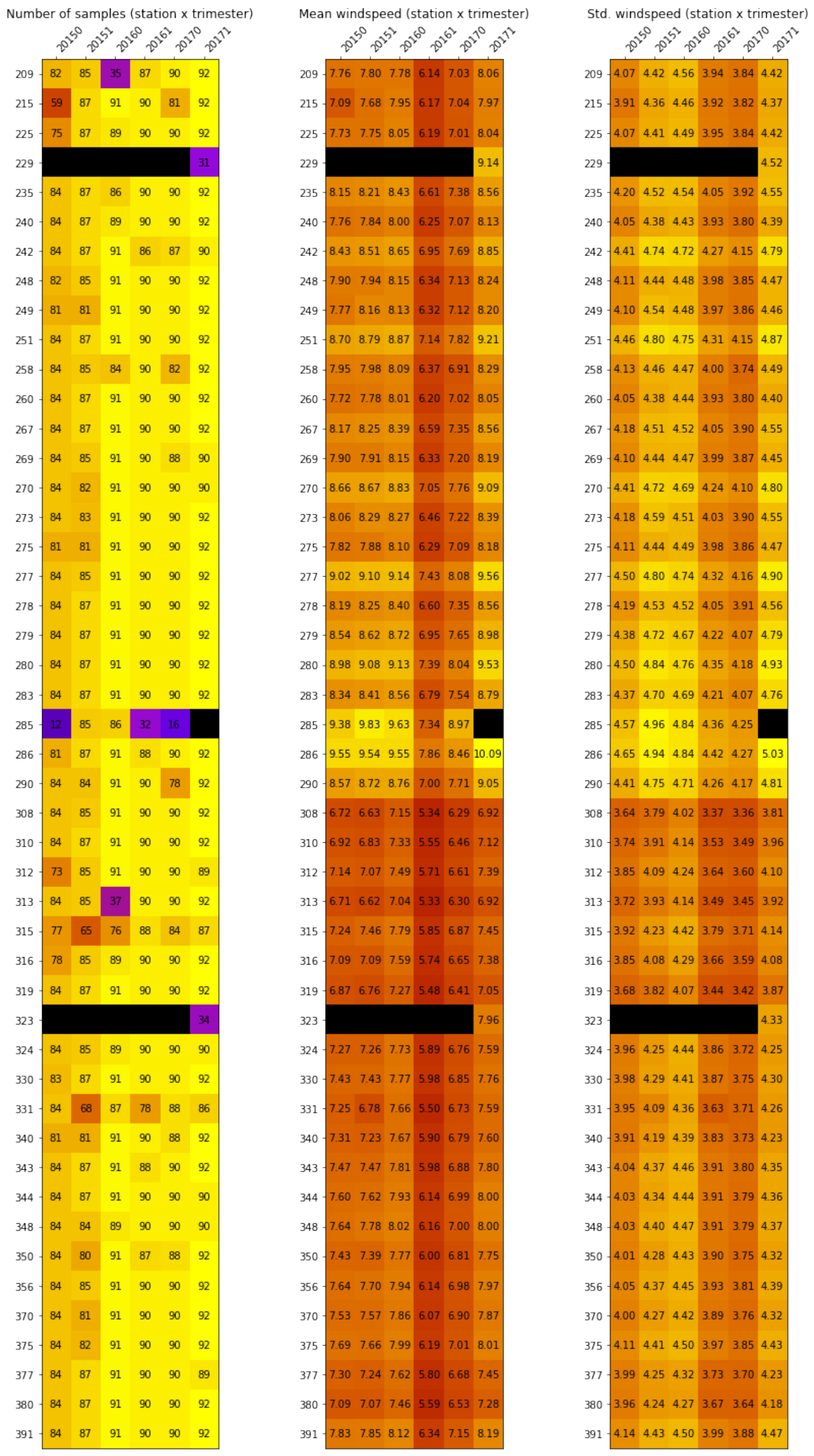
There is a clear 24h cycle in the bias and an upward standard deviation trend for almost all stations and on the global aggregate. These are intuitive features, since they match the daily cycle and the natural idea that longer lead times are harder to forecast. Assuming that the multiple time series of NWP forecast errors of each station are independent realisations of the same stochastic process, we must discard second order stationarity given that neither the mean nor standard deviation are constant over time. Notice also that each station has its own bias and standard deviation characteristics, although there are some overall similarities with higher biases during the day (lead times close to 12h and 36h) and a slight negative trend on the biases.

Moreover, we are also interested in the autocorrelation of the forecast errors in order to verify whether there is temporal persistence in them. Since stationarity was discarded by the previous observations, we cannot assume that the correlation between lead times is purely a function of the lag. Therefore, we begin by computing the aggregate correlation matrix over all stations and lead times, displayed in Figure 4.

All correlations are positive with higher values concentrated along the main diagonal, indicating that close lead times have higher correlation, as expected. The hypothesis that lead time correlations are an explicit function of the lag is equivalent to having constant values in each diagonal of the matrix. Calculating the standard deviation for each diagonal we obtain values lower than 0.045, strongly supporting the hypothesis. Hence we can meaningfully interpret the empirical temporal autocorrelations given in Figure 5.

We are most interested in lower lags, and the graphs show an exponential decay pattern for most stations. In time series analysis, this feature is characteristic of autoregressive processes (AR). AR models were used in [30, 31] for postprocessing temperature forecasts, with the intuition that forecast errors propagate through lead times.

We conclude that statistical models for this dataset must accommodate for general non-stationary processes with variable bias and standard deviation profiles while still taking into account the temporal correlation between errors.



Number of samples (station x trimester)

	20150	20151	20160	20161	20170	20171
209	82	85	35	87	90	92
215	59	87	91	90	81	92
225	75	87	89	90	90	92
229						31
235	84	87	86	90	90	92
240	84	87	89	90	90	92
242	84	87	91	86	87	90
248	82	85	91	90	90	92
249	81	81	91	90	90	92
251	84	87	91	90	90	92
258	84	85	84	90	82	92
260	84	87	91	90	90	92
267	84	87	91	90	90	92
269	84	85	91	90	88	90
270	84	82	91	90	90	90
273	84	83	91	90	90	92
275	81	81	91	90	90	92
277	84	85	91	90	90	92
278	84	87	91	90	90	92
279	84	87	91	90	90	92
280	84	87	91	90	90	92
283	84	87	91	90	90	92
285	12	85	86	32	16	
286	81	87	91	88	90	92
290	84	84	91	90	78	92
308	84	85	91	90	90	92
310	84	87	91	90	90	92
312	73	85	91	90	90	89
313	84	85	37	90	90	92
315	77	65	76	88	84	87
316	78	85	89	90	90	92
319	84	87	91	90	90	92
323						34
324	84	85	89	90	90	90
330	83	87	91	90	90	92
331	84	68	87	78	88	86
340	81	81	91	90	88	92
343	84	87	91	88	90	92
344	84	87	91	90	90	90
348	84	84	89	90	90	90
350	84	80	91	87	88	92
356	84	85	91	90	90	92
370	84	81	91	90	90	92
375	84	82	91	90	90	92
377	84	87	91	90	90	89
380	84	87	91	90	90	92
391	84	87	91	90	90	92

Mean windspeed (station x trimester)

	20150	20151	20160	20161	20170	20171
209	7.76	7.80	7.78	6.14	7.03	8.06
215	7.09	7.68	7.95	6.17	7.04	7.97
225	7.73	7.75	8.05	6.19	7.01	8.04
229						9.14
235	8.15	8.21	8.43	6.61	7.38	8.56
240	7.76	7.84	8.00	6.25	7.07	8.13
242	8.43	8.51	8.65	6.95	7.69	8.85
248	7.90	7.94	8.15	6.34	7.13	8.24
249	7.77	8.16	8.13	6.32	7.12	8.20
251	8.70	8.79	8.87	7.14	7.82	9.21
258	7.95	7.98	8.09	6.37	6.91	8.29
260	7.72	7.78	8.01	6.20	7.02	8.05
267	8.17	8.25	8.39	6.59	7.35	8.56
269	7.90	7.91	8.15	6.33	7.20	8.19
270	8.66	8.67	8.83	7.05	7.76	9.09
273	8.06	8.29	8.27	6.46	7.22	8.39
275	7.82	7.88	8.10	6.29	7.09	8.18
277	9.02	9.10	9.14	7.43	8.08	9.56
278	8.19	8.25	8.40	6.60	7.35	8.56
279	8.54	8.62	8.72	6.95	7.65	8.98
280	8.98	9.08	9.13	7.39	8.04	9.53
283	8.34	8.41	8.56	6.79	7.54	8.79
285	9.38	9.83	9.63	7.34	8.97	
286	9.55	9.54	9.55	7.86	8.46	10.09
290	8.57	8.72	8.76	7.00	7.71	9.05
308	6.72	6.63	7.15	5.34	6.29	6.92
310	6.92	6.83	7.33	5.55	6.46	7.12
312	7.14	7.07	7.49	5.71	6.61	7.39
313	6.71	6.62	7.04	5.33	6.30	6.92
315	7.24	7.46	7.79	5.85	6.87	7.45
316	7.09	7.09	7.59	5.74	6.65	7.38
319	6.87	6.76	7.27	5.48	6.41	7.05
323						7.96
324	7.27	7.26	7.73	5.89	6.76	7.59
330	7.43	7.43	7.77	5.98	6.85	7.76
331	7.25	6.78	7.66	5.50	6.73	7.59
340	7.31	7.23	7.67	5.90	6.79	7.60
343	7.47	7.47	7.81	5.98	6.88	7.80
344	7.60	7.62	7.93	6.14	6.99	8.00
348	7.64	7.78	8.02	6.16	7.00	8.00
350	7.43	7.39	7.77	6.00	6.81	7.75
356	7.64	7.70	7.94	6.14	6.98	7.97
370	7.53	7.57	7.86	6.07	6.90	7.87
375	7.69	7.66	7.99	6.19	7.01	8.01
377	7.30	7.24	7.62	5.80	6.68	7.45
380	7.09	7.07	7.46	5.59	6.53	7.28
391	7.83	7.85	8.12	6.34	7.15	8.19

Std. windspeed (station x trimester)

	20150	20151	20160	20161	20170	20171
209	4.07	4.42	4.56	3.94	3.84	4.42
215	3.91	4.36	4.46	3.92	3.82	4.37
225	4.07	4.41	4.49	3.95	3.84	4.42
229						4.52
235	4.20	4.52	4.54	4.05	3.92	4.55
240	4.05	4.38	4.43	3.93	3.80	4.39
242	4.41	4.74	4.72	4.27	4.15	4.79
248	4.11	4.44	4.48	3.98	3.85	4.47
249	4.10	4.54	4.48	3.97	3.86	4.46
251	4.46	4.80	4.75	4.31	4.15	4.87
258	4.13	4.46	4.47	4.00	3.74	4.49
260	4.05	4.38	4.44	3.93	3.80	4.40
267	4.18	4.51	4.52	4.05	3.90	4.55
269	4.10	4.44	4.47	3.99	3.87	4.45
270	4.41	4.72	4.69	4.24	4.10	4.80
273	4.18	4.59	4.51	4.03	3.90	4.55
275	4.11	4.44	4.49	3.98	3.86	4.47
277	4.50	4.80	4.74	4.32	4.16	4.90
278	4.19	4.53	4.52	4.05	3.91	4.56
279	4.38	4.72	4.67	4.22	4.07	4.79
280	4.50	4.84	4.76	4.35	4.18	4.93
283	4.37	4.70	4.69	4.21	4.07	4.76
285	4.57	4.96	4.84	4.36	4.25	
286	4.65	4.94	4.84	4.42	4.27	5.03
290	4.41	4.75	4.71	4.26	4.17	4.81
308	3.64	3.79	4.02	3.37	3.36	3.81
310	3.74	3.91	4.14	3.53	3.49	3.96
312	3.85	4.09	4.24	3.64	3.60	4.10
313	3.72	3.93	4.14	3.49	3.45	3.92
315	3.92	4.23	4.42	3.79	3.71	4.14
316	3.85	4.08	4.29	3.66	3.59	4.08
319	3.68	3.82	4.07	3.44	3.42	3.87
323						4.33
324	3.96	4.25	4.44	3.86	3.72	4.25
330	3.98	4.29	4.41	3.87	3.75	4.30
331	3.95	4.09	4.36	3.63	3.71	4.26
340	3.91	4.19	4.39	3.83	3.73	4.23
343	4.04	4.37	4.46	3.91	3.80	4.35
344	4.03	4.34	4.44	3.91	3.79	4.36
348	4.03	4.40	4.47	3.91	3.79	4.37
350	4.01	4.28	4.43	3.90	3.75	4.32
356	4.05	4.37	4.45	3.93	3.81	4.39
370	4.00	4.27	4.42	3.89	3.76	4.32
375	4.11	4.41	4.50	3.97	3.85	4.43
377	3.99	4.25	4.32	3.73	3.70	4.23
380	3.96	4.24	4.27	3.67	3.64	4.18
391	4.14	4.43	4.50	3.99	3.88	4.47

Figure 2: Training/validation dataset basic statistics per station and trimester.

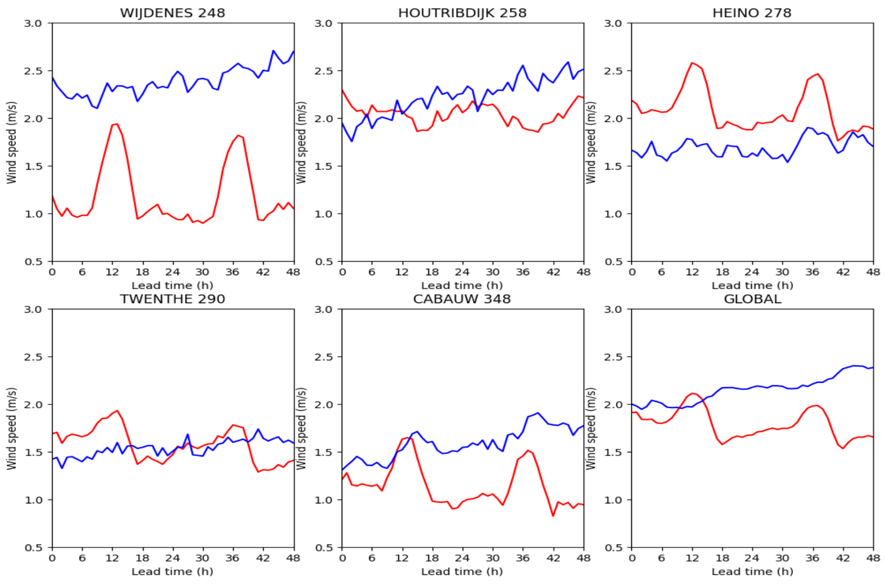


Figure 3: In red we have the bias and in blue we have the standard deviation given at each lead time for a few stations and for the global aggregate over all stations. The selection of stations presents some of the distinct bias and standard deviation profiles seen in the dataset.

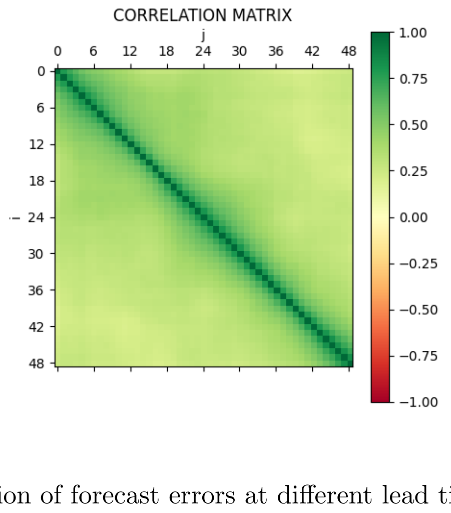


Figure 4: Correlation of forecast errors at different lead times for all stations.

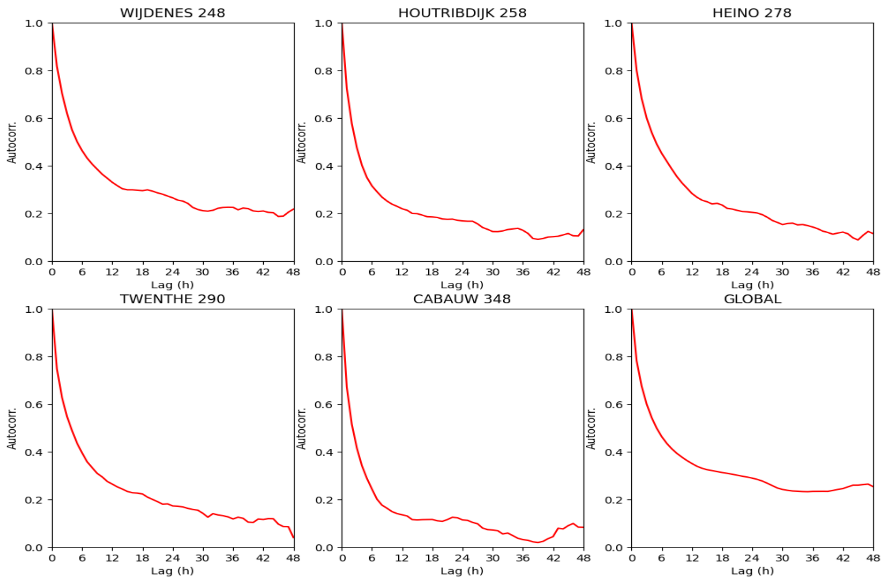


Figure 5: Autocorrelation as function of the lag for a few stations and aggregate over all stations.

3 Multivariate Postprocessing

In general, multivariate postprocessing problems don't reduce to multiple univariate problems. This only happens in the simple case that we can assume that all variables are independent. Therefore specific tools are required for issuing multivariate forecasts and evaluating their performance. In this section we present the statistical methods applied to our specific problem as well as the verification measures used to evaluate forecast quality.

First we introduce EMOS, a simple linear model used for univariate ensemble postprocessing that has been adapted to our problem. Then, we present the Schaake Shuffle (SS), an empirical copula method that aggregates a collection of marginal forecasts to produce a sample based multivariate forecast. This combination of statistical methods is referred to as EMOS/SS.

We follow up with the ARMOS(p), the first contribution of this study. This model is based on EMOS and introduces autoregressive parameters that allow forward propagation of errors through time. This introduces explicit temporal dependency in the forecast in a way that, to the best of our knowledge, has not been tried before.

Finally, we discuss the verification measures used to assess the global and marginal quality of the forecasts in the experiments.

3.1 EMOS

The Ensemble Model Output Statistics (EMOS) [15, 3] is a univariate ensemble postprocessing model that fits the parameters of a distributions (e.g. mean and variance of a normal distributions) from an ensemble of NWP predictions for a fixed lead time. Ensemble predictions are obtained from multiple runs of an NWP model with perturbed initial conditions, providing a probabilistic forecast in the form of a collection $\{f_1, \dots, f_M\} \subset \mathbb{R}$ of numerical predictions. From this ensemble, EMOS ouputs a parametric forecast distribution

$$\mathcal{D}(\mu, \sigma^2) \tag{1}$$

by estimating μ and σ^2 as a function of the ensemble values. The simplest way to do that is to use the ensemble mean $\overline{f} = \frac{1}{M}\sum_{i=1}^{M} f_i$ and the ensemble variance

$$S^2 = \frac{1}{M-1}\sum_{i=1}^{M}(f_i - \overline{f})^2 \tag{2}$$

to estimate the mean and variance of the forecast distribution using linear regression. This gives us

$$\mathcal{D}\left(a_0 + a_1\overline{f},\ b_0 + b_1 S^2\right), \tag{3}$$

where the parameters a_0, a_1, b_0, b_1 are estimated from the training data by minimizing a chosen error function. Typical distributions $\mathcal{D}$ used in postprocessing of wind speed forecasts include the Truncated Normal, Log Normal and Weibull [3, 23], but we focus on using the Truncated Normal for simplicity. It is out of the scope of this study to compare between these choices.

It is clear however that this model has to be adapted to our needs given that only deterministic NWP forecasts are available. Furthermore, it is also desirable that we introduce the spatial information from the grid of wind speed predictions and the other available variables into this model too. Hence, we let $f_1, \dots, f_M$ be the HA40 forecasts for different weather variables from Table 1 (the wind speed prediction at 10m height is naturally the most important one and should always be included), and we substitute S^2 by the spatial variance of wind speed predictions on a neighbourhood of the station, given by

$$S^2 = \frac{1}{N^2-1}\sum_{i,j=1}^{N}(w_{ij} - \overline{w})^2, \tag{4}$$

where w_{ij} are the HA40 wind speed forecasts on a (N, N) square grid with the station closest to the central grid point and $\overline{w}$ is the average of the values in the grid. In this context we cannot estimate the distribution mean using $\overline{f}$ anymore, so instead (3) is adapted to

$$\mathcal{D}(a_0 + a_1 f_1 + \dots + a_M f_M, b_0 + b_1 S^2). \tag{5}$$

This allows each weather variable f_i to contribute linearly to the estimation of the distribution mean via the a_i coefficient. Notice that it is necessary to determine which are the relevant variables and what is the ideal size of the neighbourhood around the station during model selection. With these adaptations it is possible to apply such a model to each lead time, resulting in a collection of marginal forecast distributions for each lead time according to

$$\mathcal{D}_t(\mu_t, \sigma_t^2), \ t = 0, \ldots, T \tag{6}$$

where

$$\begin{aligned} \mu_t &= a_{0,t} + a_{1,t}\ f_{1,t} + \cdots + a_{M,t}\ f_{M,t}, \quad (7)\\ \sigma_t^2 &= b_{0,t} + b_{1,t}\ S_t^2, \quad (8)\\ t &= 0, \ldots, T. \end{aligned}$$

Equations (7) to (8) define what we refer to as *linear* EMOS. The choice of linear regression for estimation of the parameters μ_t, σ_t^2 from the HA40 predictions is purely arbitrary, and we later explore non-linear alternatives such as the EMOSnet, which uses neural networks for that task. The statistical framework provided by EMOS models allows different lead times to have a characteristic bias and uncertainty, an important required feature as mentioned in Section 2.2, but does not establish any temporal dependency – the lead times are modelled independently. In order to introduce correlation between lead times we must apply an empirical copula method such as the Schaake Shuffle.

3.2 The Schaake Shuffle

The Schaake Shuffle (SS) [10] fits in the broader picture of copula methods [36]. These methods generate multivariate forecasts by reconstructing the dependency structure in a series of independent marginal forecasts using historical data. More specifically, the Schaake Shuffle applies the rank order structure from historical observations to a collection of samples from the marginal forecasts to produce more realistic multivariate samples.

Table 2 shows how this procedure is done in a hypothetical example. First we draw a fixed number of samples from the forecast distribution for each lead time and order them as in Table 2a. Then we gather the same number of historical observations and determine their rank order as displayed in parentheses in Table 2b. Finally, we reorder the original samples matching the rank order template from the historical observations to obtain the "shuffled" output in Table 2c. This way, the samples inherit the multivariate rank dependence of the selected historical observations.

	1	2	3	4
1h	3.3	3.4	3.5	4.7
2h	0.7	1.4	2.2	2.9
3h	0.0	1.9	3.2	4.4
4h	2.3	2.8	3.5	3.7
5h	1.1	3.0	4.3	5.6
6h	3.0	3.3	4.1	4.3

(a) Ordered samples

	1	2	3	4
1h	(4) 7.6	(3) 6.9	(2) 4.5	(1) 1.7
2h	(4) 7.8	(2) 2.5	(3) 5.3	(1) 2.4
3h	(4) 5.3	(2) 3.2	(3) 4.8	(1) 2.5
4h	(4) 5.5	(1) 3.5	(3) 4.8	(2) 4.2
5h	(3) 5.2	(4) 5.9	(1) 3.6	(2) 4.2
6h	(4) 6.2	(1) 3.8	(2) 4.4	(3) 5.0

(b) Ranked historical observations

	1	2	3	4
1h	4.7	3.5	3.4	3.3
2h	2.9	1.4	2.2	0.7
3h	4.4	1.9	3.2	0.0
4h	2.8	3.7	3.5	2.3
5h	4.3	5.6	1.1	3.0
6h	4.3	3.0	3.3	4.1

(c) Shuffled samples

Table 2: Schaake Shuffle step-by-step.

The first plot in Figure 6 shows the marginal forecast distributions from the linear EMOS model together with the observed wind speed values and a sample drawn from the marginal forecast distributions for 2016/02/29 at Station 370. The sample shows very erratic behaviour when compared to the observation since it does not take into account the persistence of the weather conditions through time, resulting in unrealistic hourly variations. The next plot shows 4 sorted samples and the last one is their shuffled output using the dependency template of the observations from 2016/11/12 at Station 209, 2015/11/12 at Station 215, 2015/10/28 at Station 209 and 2015/10/28 at Station 215.

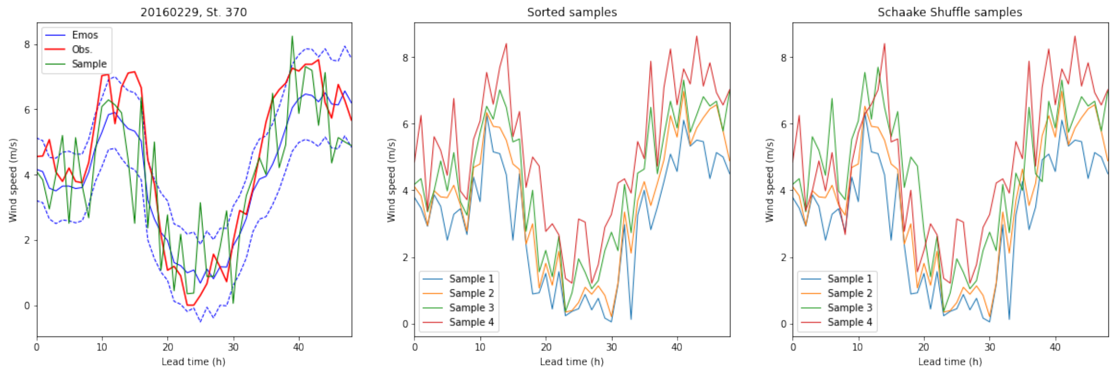


Figure 6: On the left, the mean and standard deviation interval of the EMOS marginal forecasts in blue, the observations in red and sample from the marginal forecasts in green. In the middle, four ordered samples from the forecast distribution, and on the right the shuffled output.

The quality of the output is naturally tied to the selection of these observations [10], so it is necessary to consider the procedure by which we select dates from the historical record. Ideally, these days should present conditions similar to those of the forecast, thus providing a more reliable rank order structure. For instance, the previous example could be improved by selecting historical observations from the same period in the preceding year and from Station 370. Further efforts could be made to ensure more similarity between the forecast and the selected historical observations, but we focused on restricting the selection to similar dates and to the same location.

3.3 ARMOS(p)

The ARMOS(p) model has been conceived as an alternative to the two step copula approach, which fits marginal distributions and introduces dependencies with a copula method as described previously. Taking inspiration from time series analysis methods, we propose to fit an explicit multivariate forecast distribution instead. In previous works [30, 31] it has been observed that the error series of NWP temperature forecasts present autoregressive behaviour, that is, the forecast errors $Z_t = W_t - w_t$ at time t, where W_t is the weather variable and w_t is the NWP forecast, depend directly on previous errors via the formula

$$Z_t - \mu = \phi_1(Z_{t-1} - \mu) + \cdots + \phi_p(Z_{t-p} - \mu) + \epsilon_t, \ t \geq p \tag{9}$$

for some $p \geq 0$, where $\mathbb{E}[Z_t] = \mu$ and $\epsilon_t \sim N(0, \sigma^2)$ are i.i.d. For the initial values $t < p$ we assume $Z_t = \mu + \epsilon_t$. Some key features of such autoregressive models are second order stationarity (provided the coefficients ϕ_j have some adequate behaviour) and their exponentially decreasing autocorrelation profile.

Assuming the forecast errors satisfy the AR property as in (9), we observe that

$$W_t = (w_t + \mu) + \sum_{j=1}^{p} \phi_j(W_{t-j} - w_{t-j} - \mu) + \epsilon_t, \tag{10}$$

and for the initial values $t < p$ we have $W_t = (w_t + \mu) + \epsilon_t$. Hence the AR hypothesis for the errors is equivalent to the above statistical behaviour of the observed wind speeds. In particular we have $\mathbb{E}[W_t] = w_t + \mu$.

This approach however clearly does not match the properties of forecast errors observed in Section 2. First, we know that the bias and deviation in the forecast errors present different behaviours at different lead times, with a clear daily cycle. Additionally, in the stochastic process above W_t is always normally distributed, which is not suitable for wind speed predictions because it allows negative speeds. We address these issues in the following way.

First, we must allow for distinct mean and variance coefficients for each lead time. Hence the equations above take the new form

$$W_t = (w_t + \mu_t) + \sum_{j=1}^{p} \phi_j (W_{t-j} - w_{t-j} - \mu_{t-j}) + \epsilon_t, \tag{11}$$

with $\epsilon_t \sim N(0, \sigma_t^2)$ independent. This way we have $\mathbb{E}[W_t] = w_t + \mu_t$, allowing each lead time to have its own bias characteristic. The variance does not have a similar easy closed form. However, since ϵ_t is independent of W_i for $i < t$ we know from (11) that $\mathbb{V}[W_t] > \sigma_t^2$, which can be interpreted as lead time t "inheriting" uncertainty from the previous lead times via the autoregressive property. In the simple case $p = 1$ we can write $\phi = \phi_1$ and by induction obtain

$$\mathbb{V}[W_t] = \sum_{i=0}^{t} \phi^{2i} \sigma_{t-i}^2. \tag{12}$$

Another way to interpret (11) is is in terms of conditional distributions. Indeed, we can define the joint distribution of the entire process by conditioning the distribution W_t on its previous values, which gives us

$$W_t | W_{t-1}, \dots, W_0 \sim N\left(w_t + \mu_t + \sum_{j=1}^{p} \phi_j (W_{t-j} - w_{t-j} - \mu_{t-j}),\ \sigma_t^2 \right). \tag{13}$$

Although more cumbersome, this equivalent formulation of the process allows us to consider alternative distributions $\mathcal{D}$ other than the Normal as below. In order to ensure positivity of the process we use the Truncated Normal (values in $[0, \infty)$) as we did for EMOS. Finally, we let $\hat{\mu}_t := w_t + \mu_t$ to obtain the ARMOS(p) model

$$W_t | W_{t-1}, \dots, W_0 \sim \mathcal{D}_t \left(\hat{\mu}_t + \sum_{j=1}^{p} \phi_j (W_{t-j} - \hat{\mu}_{t-j}),\ \sigma_t^2 \right) \tag{14}$$

with coefficients $\hat{\mu}_t,\ \sigma_t^2,\ t = 1, \dots, T$ and $\phi_j,\ j = 1, \dots, p$. From now own we change notation $\mu_t := \hat{\mu}_t$ for simplicity whenever we refer to the ARMOS(p) model.

In practice we estimate these coefficients from data, by either letting them fixed or by considering them as a function of the input weather variables. If we take

$$\mu_t = a_{0,t} + a_{1,t}\ f_{1,t} + \dots + a_{M,t}\ f_{M,t} \tag{15}$$

$$\sigma_t^2 = b_{0,t} + b_{1,t}\ S_t^2 \tag{16}$$

$$\phi_j = 0,\ (j = 1, \dots, p) \tag{17}$$

where $\{f_{i,t}\}$ are the HA40 variables we get the standard EMOS (without Schaake Shuffle). So ARMOS(p) is a strict generalization of EMOS, justifying its name. If instead of forcing $\phi_j = 0$ we allow the autoregressive to be freely trainable, then we obtain the *linear* ARMOS(p) model. Like EMOS, the ARMOS(p) also allows for alternative parameter estimation methods, and we explore the usage of neural networks to obtain the ARMOS(p)net.

Figure 7 shows an example of the predictions of the linear ARMOS(1) by displaying conditional forecast distributions. Before the vertical line at the 24h lead time we see one step ahead predictions where we "know" the observation at the previous lead time. After the vertical line we stop updating the observations, so we obtain the conditional forecast with knowledge up to the 23h lead time. Notice that the AR correction provides a good fit for the one step ahead predictions, and for the n-step ahead predictions the variance is increased to take into account the accumulation of uncertainty through lead times.

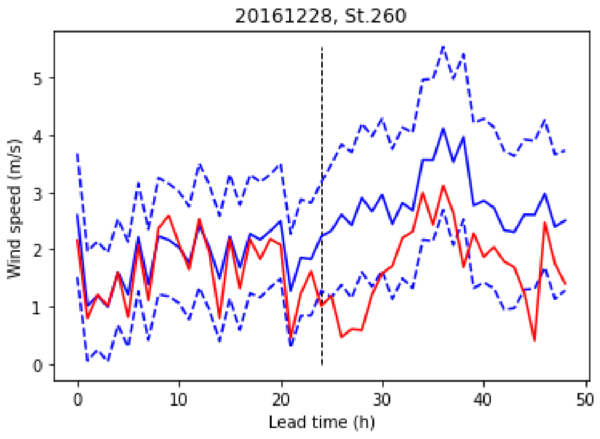


Figure 7: Linear ARMOS(1) predictions for 2016/12/28 at Station 260. In this case $\phi = 0.649$ and in red we see the observation, the solid blue line is the mean one step ahead predictions until the 24h lead time, after that it shows the mean n-step ahead prediction starting at the 23h lead time. The dashed blue line show the interval of one standard deviation.

3.4 Proper Scoring Rules

This section follows Bjerregård et al. [6].

Scoring rules are summary measures that allow quantitative comparisons between different probabilistic forecasts. They quantify the *error* or *loss* of the forecast and are used for fitting statistical models to training data. This is done by applying minimization algorithms that select the optimal model parameters that ensure minimal average loss over the training samples. Thus, scoring rules assign a numerical value $S(\mathcal{D}, y)$ to any pair $(\mathcal{D}, y)$ where $\mathcal{D}$ is a forecast distribution for some variable Y and y is an observation of that variable. A fundamental property that every desirable scoring rule must satisfy is that it must be *proper* [14, 37]. Essentially, a scoring rule is proper if it is "unbiased", ensuring that the true distribution of Y is always optimal with respect to it. More precisely, given that Y has distribution $\mathcal{F}$, we say that a scoring rule S is proper if for any forecast distribution $\mathcal{D}$ we have

$$\mathbb{E}_Y[S(\mathcal{F}, Y)] \leq \mathbb{E}_Y[S(\mathcal{D}, Y)]. \tag{18}$$

If the inequality is strict whenever $\mathcal{D} \neq \mathcal{F}$ we say that the scoring rule is *strictly* proper. The use of non-proper scoring rules should be avoided, since they allow the possibility that a forecast distributions could strictly outperform the real distribution, leading to misguided inferences. Furthermore, in order to fit a model to data by minimizing some scoring rule we must only consider strictly proper scoring rules, since those are the only ones that ensure that the distribution with optimal scoring is indeed the true distribution of Y. We consider three multivariate proper scoring rules: the Logarithmic Score [16], the Energy Score [14] and the Variogram Score [37]. Each of them receives the forecast distribution $\mathcal{D}$ either in the form of its PDF or CDF.

The Logarithmic Score (LS) is nothing but the negative log-likelihood, which is widely used in statistics for Maximum Likelihood parameter estimation. Given that P is the PDF of the forecast distribution $\mathcal{D}$ and $y \in \mathbb{R}$ is the observed value, the LS is defined as

$$\text{LS}(\mathcal{D}, y) = -\log P(y). \tag{19}$$

The LS is a strictly proper scoring rule, it can be applied in a vast range of problems, both univariate and multivariate, and it penalizes for the marginal calibration and correlation structure simultaneously. It can also be computed analytically for both EMOS and ARMOS by decomposing it with respect to conditional distributions as shown below.

$$\text{LS}(\mathcal{D}, y) = -\sum_{t=0}^{T} \log P(y_t | y_{t-1}, \ldots, y_0) \tag{20}$$

For applications in the EMOS and ARMOS models, we use the PDF of a Truncated Normal with range $[0, \infty)$ to obtain the LS as a function of the parameters $\mu_t, \sigma_t^2, \phi_j$. Given the distribution parameters μ, σ^2 the PDF is given by $P(y|\mu, \sigma^2) = \frac{\frac{1}{\sigma}\ \varphi\left(\frac{y-\mu}{\sigma}\right)}{1-\Phi\left(\frac{y-\mu}{\sigma}\right)}$ where φ and Φ are the Standard Normal PDF and CDF respectively. In the case of EMOS, (6) gives a series of independent Truncated Normal distributions that allows us to compute the LS of $y \in \mathbb{R}^T$ with respect to the distribution parameteres μ_t, σ_t, leading to

$$\mathrm{LS}_{\mathrm{EMOS}}(y|\mu, \sigma^2) = -\sum_{t=0}^{T} \log P_t(y_t|\mu_t, \sigma_t) \tag{21}$$

$$= -\sum_{t=0}^{T} \log \left(\frac{\frac{1}{\sigma_t}\ \varphi\left(\frac{y_t-\mu_t}{\sigma_t}\right)}{1-\Phi\left(\frac{y_t-\mu_t}{\sigma_t}\right)} \right) \tag{22}$$

For ARMOS we have a similar result by using (14) :

$$\mathrm{LS}_{\mathrm{ARMOS}(p)}(y|\mu, \sigma^2, \phi) = -\sum_{t=0}^{T} \log P_t(y_t|y_i, \mu, \sigma, \phi,\ i < t) \tag{23}$$

$$= -\sum_{t=0}^{T} \log \left(\frac{\frac{1}{\sigma_t}\ \varphi\left(\hat{\epsilon}_t\right)}{1-\Phi\left(\hat{\epsilon}_t\right)} \right), \tag{24}$$

where $\hat{\epsilon}_t := \frac{(y_t-\mu_t)-\sum_{j=1}^{p}\phi_j(y_{t-j}-\mu_{t-j})}{\sigma_t}$. The parameters μ_t, σ_t, ϕ_t are estimated from data either by linear regression, such as in the linear EMOS/ARMOS, or by alternative models such neural networks. In both cases we estimate the parameters (regression coefficients or the weights of the network) by minimizing the average LS over the training dataset. The above functions are differentiable, so the dependency of the LS on the model parameters is also differentiable. This allows for the application of (stochastic) gradient descent methods for obtaining the optimal coefficients/weights. The gradients can be obtained analytically, but in practice automatic differentiation algorithms were used [4]. The major downside of the LS is that it has been found to be sensitive to outliers in training data when fitting a model [14] since it heavily penalizes unlikely observations.

Next, we consider the Energy Score (ES), which is a multivariate generalization of the CRPS scoring rule for univariate forecasting [28]. Given F the CDF of the forecast distribution $\mathcal{D}$ and $y \in \mathbb{R}$ the observed value, the CRPS is defined as

$$\mathrm{CRPS}(\mathcal{D}, y) = \int_{\mathbb{R}} (F - \mathbb{1}_{[y,\infty)})^2 dx. \tag{25}$$

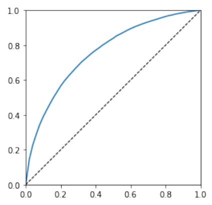
An equivalent formulation of the CRPS is given in [14]:

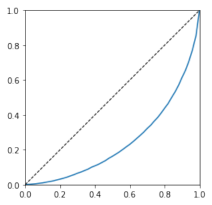
$$\mathrm{CRPS}(\mathcal{D}, y) = \mathbb{E}|X - y| - \frac{1}{2}\mathbb{E}|X - X'|, \tag{26}$$

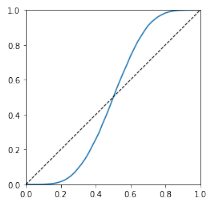
where $X, X' \sim \mathcal{D}$ are independent random variables. In general it is hard to find analytical solutions to the integral formulation in (25) as a function of the parameters of some distribution family, so this alternative formulation is useful for obtaining good approximations by taking samples from $\mathcal{D}$. A second benefit from this formulation is that it is easily generalizable to multivariate problems by considering the Euclidean norm as a replacement for the absolute value. This leads to the definition of the Energy Score of order $\beta \in (0, 2)$:

$$\mathrm{ES}_\beta(\mathcal{D}, y) = \mathbb{E}||X - y||_2^\beta - \frac{1}{2}\mathbb{E}||X - X'||_2^\beta, \tag{27}$$

where $X, X' \sim \mathcal{D}$ are independent random vectors. In this study we only use $\beta = 1$, so from now on we omit the parameter β. The ES is a strictly proper scoring rule, but it has been found to be insensitive to the correlation structure between variables [33], penalizing mostly just for marginal miscalibration. It is also less sensitive to outliers than the LS. Unlike the LS, it is not easy to obtain an analytical expression for the ES via (27), making it necessary to rely on sampling.


(a) Positive bias


(b) Negative bias


(c) Underconfidence

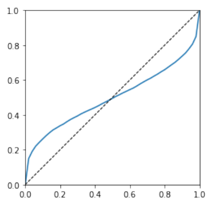

(d) Overconfidence

Figure 8: PIT diagram interpretations

The ES has found a lot of applications in multivariate weather forecasting, but due to its limitations in identifying correlation structures another score has been put forward. The Variogram Score (VS) was conceived as a purely correlational non-strict proper scoring rule. It is based on the pairwise difference between variables, thus presenting translation invariance. This way the VS does not penalize marginal calibration and therefore cannot be used on its own for the analysis of forecast quality. The Variogram Score of order $\gamma > 0$ is defined as

$$\mathrm{VS}_\gamma(\mathcal{D}, y) = \sum_{0 \le i < j \le T} w_{ij} \left(|y_i - y_j|^\gamma - \mathbb{E}|X_i - X_j|^\gamma \right)^2, \tag{28}$$

where $X \sim \mathcal{D}$ is a random vector and w_{ij} are weights. We have restricted ourselves to $\gamma = 0.5$ and to diadic weighting $w_{ij} = 2^{1-|i-j|}$ since we are mostly interested in short time lag correlations, so we omit any mentions to those from here on. Just like the ES, the VS does not have an analytical form, so it must be calculated by sampling.

3.5 Marginal Verification

This section follows Wilks [42].

Although the three scoring rules provided in the previous section give comprehensive information about forecast performance, we are also interested in evaluating marginal performance. In particular, we are specially interested in forecasting extreme values since high winds speeds can cause severe damage. Here we consider the PIT diagram for qualitative analysis of marginal forecasts and two threshold based methods, the Brier Score and Reliability diagrams.

The Probability Integral Transform (PIT) diagram is the cumulative version of the rank histogram [18], it gives a graphical representation of the quality of forecast distribution. This method is based on the well known fact that for any continuous random variable X with CDF given by F we have

$$F(X) \sim U[0, 1]. \tag{29}$$

This way, if we consider F to be a forecast distribution for some weather variable X, we may try to assess the quality of the forecast by observing how much $F(X)$ differs from a uniform distribution. Hence the PIT diagram is obtained by plotting, on the $[0, 1]$ range, the estimate CDF of $F(X)$ calculated from data. For the true distribution of X this plot should give a straight line with angular coefficient 1, so we evaluate the quality of the forecast by the deviation from the main diagonal. Figure 8 shows the possible interpretations of some PIT plots.

Another tool for marginal validation is the Brier Score (BS). It is a verification measure for probabilistic forecasts with binary outcome. That is, if $p \in [0, 1]$ is the forecast probability that a variable

$y \in \{0, 1\}$ is equal to 1, then the Brier Score is given by

$$\mathrm{BS}(p, y) = (p - y)^2. \tag{30}$$

Notice that the average value of the BS over a dataset is nothing but the mean squared error of the forecasts. In the context of wind speed forecasts, it is necessary to convert the problem into a binary classification setting in order to apply the BS. This is done by establishing a threshold $H > 0$ and a fixed lead time, observations that exceed the threshold at that lead time are labeled 1 and the others 0. The probability forecast is simply $1 - F(H)$, where F is the marginal forecast CDF for that particular lead time. When comparing the performance of two models, the BS is often presented in the format of a *relative skill score*, which we refer to as the Brier Skill Score (BSS). Given $BS_1, BS_2 \geq 0$ the BS of models 1 and 2, the BSS of model 1 with respect to model 2 is defined as

$$BSS = 1 - \frac{BS_1}{BS_2}. \tag{31}$$

Positive values of the BSS indicate that model 1 outperforms model 2, and the contrary for negative values.

Finally, we consider the Reliability diagram. Like the BS, this method also relies on reducing the problem to binary classification using a wind speed threshold $H > 0$. However, this time we are interested in verifying reliability, which measures how often we observe a positive outcome given the forecast $p \in [0, 1]$. In an ideal scenario, the observed frequency of the positive outcome in the group of observations with forecasted probability p should be equal to p. The Reliability diagram is obtained by selecting a few values of p and plotting the value of p against the observed frequency of positive outcomes. Like in the PIT diagram, we can evaluate forecast reliability by observing the deviation of the curve from the main diagonal. In general, reliability curves below/above the diagonal represent over/under forecasting, and slope smaller/larger than 1 represent over/under confidence.

Extreme events are evaluated by considering high thresholds H. Naturally these extreme events are very rare, and for most samples the forecast probability p will be very low. Hence it is also important to display, in addition to a Reliability diagram, the frequencies of each forecast probability. This is the so called Refinement diagram, since it shows how the model outputs are distributed over the interval $[0, 1]$. Ideally, the outputs should concentrate in the extremes of the interval, meaning that in most cases the model can confidently distinguish between positive and negative outcomes.

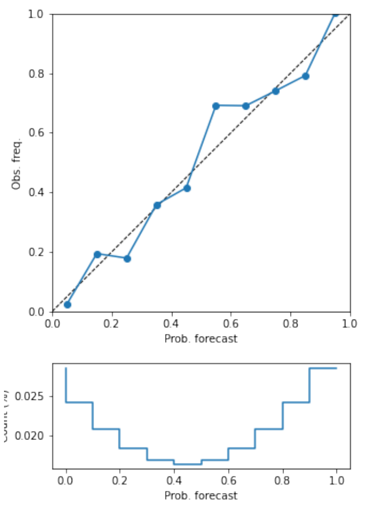


Figure 9: Reliability diagram example.

4 Deep Learning

This section follows Goodfellow et al. [17] and Yu et al. [43].

The field of Deep Learning has seen unprecedented developments in recent years. Even though perceptrons, the first conceptual step towards the modern concept of a neural network, were conceived in the end of the 1950s [34], only with recent developments in computational technology they have found widespread applicability. These networks have been used to solve a wide range of problems both in industry and academia including classification, regression, object detection, natural language processing, and noise filtering problems as a few examples.

In general, neural network models can be massive in size containing up to millions of parameters. They are built from a basic building block called a "neuron", which is essentially a generalized linear model on its own, having linear parameters for the input and a non-linear activation function to generate the output. The unprecedented versatility of neural networks comes from using several neurons in parallel to form a *layer* and stacking consecutive layers.

Several variants of this basic structure have been proposed in order to tackle specific problems. Previous works [41] have successfully applied convolutional networks to the problem of issuing a forecast PDF for wind speed by estimating the parameters μ, σ^2 of a truncated normal distribution from the HA40 predictions. In our context it is possible to do something equivalent for both the EMOS and ARMOS models requiring the network to predict the parameters μ_t, σ_t^2 for each lead time as well as the ϕ_j in the case of ARMOS.

In this section we provide an overview of the neural network architectures used for parameter estimation of the EMOS and ARMOS models. We focus the discussion to the topics of convolutional, LSTM and Wide & Deep architectures. It is not our goal to provide any further background on Deep Learning – for that we recommend [17].

4.1 Convolutional Layers

Problems involving image processing presented serious challenges for early neural network models. Image data is very high dimensional, possibly with three numerical values (RGB) for each pixel in a 1000×1000 grid, and information is localized in the sense that only groups of neighbouring pixels are meaningfully related, the relationship between faraway pixels is mostly irrelevant. Furthermore, a given feature (e.g. a particular object) should be identified in any placement in the image, so learning must be shift invariant.

These characteristics explain why purely dense neural networks performed so poorly in problems involving images. Fully connecting the neurons to every pixel generated enormous complexity with millions of parameters. Furthermore, several of these connections would have to represent redundant information, since the network would have to relearn how to recognize a pattern for every change in position. Thee lack of locality in dense models also means that these models were very prone to overfitting, identifying false patterns from spread out collections of pixels. The convolutional architecture developed by [25] addressed all these issues and soon became the state-of-the-art method for a variety of image processing problems such as image classification or object detection.

Convolutional networks solve the issues described above by introducing the *convolutional layer*. This layer can be thought of as a set of *filters* (or *kernels*) that are applied to the image to output *feature maps*. Given a bi-dimensional input $I \in \mathbb{R}^{n \times n}$ and a filter $K \in \mathbb{R}^{d \times d}$, the bi-dimensional feature map S is obtained via discrete convolution and application of an activation function $\sigma : \mathbb{R} \to \mathbb{R}$, e.g. the ReLU, as in the equation below

$$S(i, j) = \sigma(K * I)(i, j) = \sigma\left(\sum_{m,n} I(i - m, j - n)K(m, n)\right). \tag{32}$$

During training the network optimizes the weights $K(m, n)$ on each filter so they can recognize distinct features in the image, such as horizontal and vertical lines or any particular shape. The "presence" of such feature in any location of the image is then registered in the feature map by the convolution with the filter. Note however that this is heuristic, in practice the identified features may not be interpretable. These feature maps can be passed to a subsequent convolutional layer or

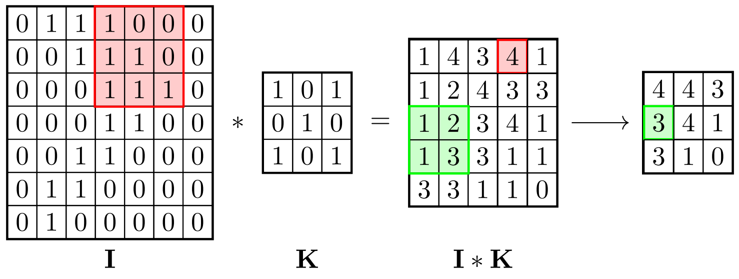


Figure 10: Discrete convolution of input I and filter K followed by 2×2 max-pooling. In this case we did not apply any activation function.

serialized and passed to a dense layer that generates the final output. Often an intermediary pooling layer is included at the end of the convolutional layers, returning the maximum value of every 2×2 block of the feature map. This is used to reduce the dimensions of the feature maps, keeping only the most important identified features. Figure 10 shows an example of discrete convolution and 2×2 max-pooling. Notice that the discrete convolution reduces the dimension of the input. To prevent that we apply padding, in which we add zeros to the borders of the input to ensure that the convolution with the filter won't reduce the dimensions.

Another way to interpret this procedure is to view each value of the feature map as the output of a single neuron. In this perspective the neuron is connected only to a small region of the image called its *perceptive field*, and its weights are precisely the weights of the filter that generated that feature map. This way all neurons in a feature map share the same weights. Effectively, the convolutional layer restricts the connections of a neuron and forces parameter sharing. This solves all the issues mentioned previously, controlling the high dimensionality of the input, reducing overfitting by imposing weight constraints and introducing the flexibility to identify patterns regardless of their positioning in the image.

In [41] convolutional networks were successfully used for postprocessing of wind speed forecasts from the Harmonie-Arome model. Instead of using only information on the grid point closest to the target station, they applied a convolutional model to the wind speed prediction in a square nighbourhood of the station, using the spatial information to improve the forecast. Given its success we decided to apply the same method and adapt it to our situation. Notice however that our problem also includes the temporal dimension, with each lead time having its own bias and dispersion characteristics. It is not feasible to apply a distinct convolutional model for each lead time, so instead we used a single convolutional layer that applies the *same* filters to all lead times. This way we apply parameter sharing throughout lead times, which forces the convolutional layer to identify purely spatial patterns and leaving the identification of temporal characteristics of the errors for further layers.

4.2 Long-Short Term Memory

Another class of problems that requires special attention are those with sequential input. This is the case with data that includes a temporal dimension in which the relationship between different times is relevant. Applying the same model to each time, as we previously considered doing with the convolutional layers, disregards all temporal infomation from the input by treating all times equally. On the other hand, developing independent models is not viable either, specially in cases that the size of temporal dimension might vary, as in the case of speech recognition of a sentence.

Recurrent neural networks (RNNs) [35] were developed in order to deal with this kind of sequential input. In particular, we focus on the LSTM architecture [20]. The LSTM cell (or layer) has a component called the *cell state* that, after each step of the sequential prediction, is passed back to the network as input via a "self-loop" together with the previous output of the cell. This state encodes the memory of the cell, and it can be changed via the *input and forget gates*. This allows the network to store information and propagate it through time. Another crucial feature of the LSTM is the *output gate*, which "decides" what parts fo the cell state will be sent as output. This is important since it allows the network to change its output without changing the cell state, preserving its memory. The LSTM architecture has found huge success in problems such as speech recognition, text translation and time series forecasting.

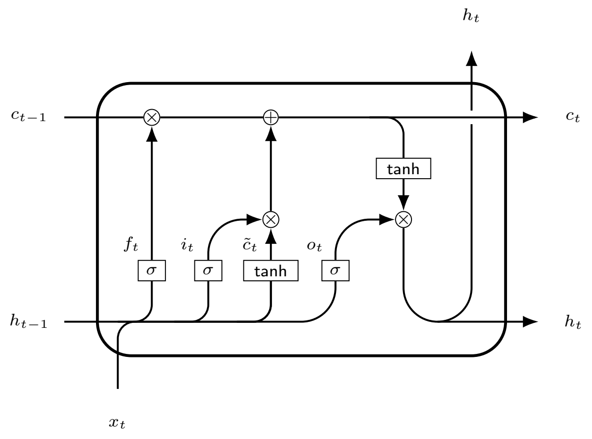


Figure 11: Representation of LSTM cell given in (33 - 38).

More explicitly, let n be the size of the input of the LSTM cell and d the number of *units* or neurons of the cell. If $x_t \in \mathbb{R}^n$ is the input at time t and $c_{t-1}, h_{t-1} \in \mathbb{R}^d$ are the are the state and output of the cell from the previous time, we have

$$f_t = \sigma(W_f \cdot [x_t, h_{t-1}] + b_f) \tag{33}$$
$$i_t = \sigma(W_i \cdot [x_t, h_{t-1}] + b_i) \tag{34}$$
$$\tilde{c}_t = \tanh(W_c \cdot [x_t, h_{t-1}] + b_c) \tag{35}$$
$$o_t = \sigma(W_o \cdot [x_t, h_{t-1}] + b_o) \tag{36}$$
$$c_t = f_t * c_{t-1} + i_t * \tilde{c}_t \tag{37}$$
$$h_t = o_t * \tanh(c_t), \tag{38}$$

where f_t, $i_t, o_t \in \mathbb{R}^d$ are the states of the forget, input and output gates, $\tilde{c}_t \in \mathbb{R}^d$ is the update to the cell state and $h_t \in \mathbb{R}^d$ is the output of the cell. In the notation above, σ is the sigmoid function, $*$ is the entrywise product and $[\cdot, \cdot]$ denotes concatenation. The σ and tanh functions are applied entrywise. Furthermore, $W_f, W_i, W_c, W_o \in \mathbb{R}^{d\times(n+d)}$ are the learnable weights of the cell associated to each of its states.

The LSTM architecture has already been applied to postprocessing in previous work, such as in [2, 38] but to the best of our knowledge it hasn't been used in postprocessing of wind speeds. After serializing the feature maps from the convolutional layer, we pass it as input to the LSTM cell which takes into account the lead time differences between marginal forecasts.

4.3 Wide & Deep

The usage of non-linear activation functions between layers of a neural network is at the heart of the predictive power of these models. The universal convergence theorem shows that this non-linearity allows the networks to learn very complex relationships in a broad functional space [21]. However, in the case in which there exists a very clear linear trend in the data, a deep stack of non-linear layers can introduce distortions and have trouble to learn this very simple behaviour.

In [8] researchers experimented with an approach that aims to fix this in the context of recommender systems. They proposed feeding the input into two paths along the network, one that was shallow with a single layer and large number of neurons, and another that was deep with several stacked layers. The outputs of each of these parts is then aggregated to generate the final output of the model. We use this idea to develop a similar model, as given in Figure 12. The main difference between the approach in [8] and ours is that in our case we don't include a shallow processing layer, we just pass the input to the output layer. This way the output layer can directly infer the linear relationship between the input and output without any distortions from non-linear activation functions, while the deep path can infer non-linear deviations from this linear relationship.

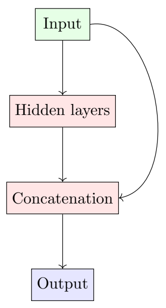


Figure 12: Wide & Deep architecture

It was observed in [41] that convolutional networks for windspeed postprocessing had best perfomance not when forecasting directly for wind speed, but when they forecasted the residuals of a linear regression instead. This is strong evidence that there is indeed a linear relationship between the HA40 wind speed forecasts and the observations, while the non-linear spatial information extracted from the network is more helpful in explaining the deviation from this linear relationship. Instead of transforming the target variable via linear regression, we decided to make use of this adaptation of the Wide & Deep architecture to ensure that the network learns this simple linear relationship.

4.4 Technical Considerations

Training neural networks is a delicate task, since the large number of parameters makes the model vulnerable to overfitting. Also, the application of gradient descent techniques introduces fundamental problems regarding convergence to an optimum. In fact, there are no theoretical guarantees that a model will converge to an optimal solution due to the irregularity of the optimization profile. This means that gradient descent can get stuck in a local minimum at any time and we are unknowingly left with a suboptimal model. Another problem involving gradient descent refers to exploding/vanishing gradients, in which learning fails due to either too large or too small gradients. Finally, there is still the issue of hyperparameter selection, since there is no way to optimally select the number of layers, filters etc. The methods designed to solve these issues are all empirical in nature, leaving us again without any guarantees.

Two very common methods that can be used to prevent overfitting are regularization techniques and early stopping [17]. The first involves adding a penalty metric to the loss function that restricts the behaviour of the weights. A standard example is ℓ_2-regularization, in which we multiply the squared ℓ_2 norm of the weights (sum of squares of weights) by a constant factor and add to the loss function. The constant factor encodes the strength of the regularization. This forces the network to optimize its output while keeping the weights as close to 0 as possible. An alternative is to apply early stopping. Instead of training the model thoroughly until convergence in the test set, it is sometimes beneficial to stop training early at a point in which the model generalizes better. This can be done by keeping track of the performance of the model in the validation set, and taking the model that performs best on the unseen data.

For gradient related problems it is possible to apply a variety of modifications to vanilla stochastic gradient descent. One of the most famous is Adaptive Moment Estimation (Adam) [24]. Exploding/vanishing gradients can be solved with input normalization, batch normalization [22], and with the application of ReLU activation functions [11] instead of sigmoids in both dense and convolutional layers. Ultimately, there is still no guarantee that the models will not converge poorly, so to avoid this uncertainty it is important to train the same architecture multiple times, randomly reinitializing the weights at each trial. This provides a more confident estimate of the performance of a given architecture.

Finally, there exist many hyperparameter selection frameworks. Most are based on trial and error, randomly selecting hyperameter configurations, training the models and comparing performance. The procedure is extremely expensive from a computational viewpoint so alternatives to simple random selection have been proposed. One of those is to apply Bayesian Optimization [29, 40] to the process, biasing the random selection of hyperparameters towards the best performant features. This heuristic "zooms" into the best configurations, resulting in a more efficient exploration of the configuration space.

5 Experiments

5.1 Methods

The conducted experiments aim to compare the performance of the EMOS/SS and ARMOS statistical models, as well as how each of them benefits from the application of neural networks for the estimation of the parameters. In this context, we introduce temporal dependency into the models in two ways: explicitly in the statistical model via the autoregressive parameters of the ARMOS(p) model, or in the parameter estimation via neural networks using the LSTM architecture. Notice that these are not exclusive, since it is possible to estimate the ARMOS mean and variance parameters via an LSTM path while the global autoregressive parameters come from dense layers, since they do not display sequential structure.

An important matter in this context is how to define our benchmark. Indeed, neural networks have enormous predictive capabilities and can have millions of parameters, it would not be fair to simply compare a neural network with a linear model. In fact, it is common practice in postprocessing to apply the linear EMOS/SS framework *locally*, fitting one model for each station, as opposed to *globally*, one model fitted to all stations simultaneously. This naturally increases the predictive power of the model, since the local models can adapt to the error characteristics of the stations. Indeed Figure 3 in Section 2 shows how much these patterns may vary for each location. The same can be said about linear ARMOS. On the other hand, neural networks always have to be fitted globally, because there is not enough data per individual station for training and it would be too costly to run one model per station.

Hence, we distinguish between three types of EMOS and ARMOS models: the network models, the local linear models and the global linear models. Both the network models and global linear models predict globally, so they perform the same task, but the former are much more complex when compared to the latter. Meanwhile, the local linear models are an intermediary benchmark, they have more complexity than the global linear models (roughly 45 times the number of parameters) and predict locally, which slightly modifies the prediction task. Thus, to obtain a fair benchmark for the neural network models we must compare them with these two groups of linear models. In each of the three groups we consider the EMOS/SS and ARMOS(p) with $p = 1, 2, 3$. In the case of neural networks we further distinguish between the EMOS/SS with or without the use of the LSTM architecture and obtain the EMOSnet/SS and LSTM/EMOSnet/SS. Furthermore, even though the temporal dependencies are modeled explicitly, it is still possible to apply the Schaake Shuffle to the samples of ARMOS. This destroys the formal statistical relationship between lead times, but we include it in the experiments nonetheless to verify if it is empirically beneficial.

The comparison between all these variants should provide a comprehensive picture of the benefits and disadvantages of each modelling option. We evaluate the performance of each of these models by calculating the average Logarithmic, Energy and Variogram Scores. Marginal validation is also performed using the Brier Score together with the PIT and Reliability diagrams. All validation metrics are computed on the test dataset, which was previously unseen by the models.

5.2 Model Description and Selection

The linear EMOS models are determined by equations (6) to (8) from Section 3, and linear ARMOS is given by equations (14) - (17). From the variables in Table 1, we disregard the wind speeds at 850 and 925 hPa since those are too correlated with the main independent variable – wind speed predictions at 10m height. Thus the models forecast for 49 consecutive lead times (0h - 48h) and use 5 NWP predictor variables: wind speed at 10m as the main variable and mean sea level pressure, turbulent kinetic energy, humidity at surface level and geopotential height at 500 hPa as auxiliary variables.

The size of the neighbourhood used for calculation of the spatial variance S^2 was determined by estimating global linear models using only the wind speed at 10m as predictor. Models were trained for square neighbourhood sizes which are multiples of 5 ranging from 5 up to 50. In this case the neighbourhood of size x is a $(2x + 1) \times (2x + 1)$ square grid with the station closest to the central grid point. After establishing the optimal grid, forward selection was used to select the best variables. For local models this was performed for every station and then selecting the variables and grid size that were optimal for most models to ensure the same hyperparameters were used for all stations.

The network models use components from each part of Section 4, including convolutional, dense and

LSTM components to integrate spatial and temporal information. Figure 13a provides a general visual representation of the network topology. The convolutional blocks are each composed of convolutional, batch normalization and 2×2 max pooling layers as displayed in Figure 13b. The same convolutional blocks are applied to all lead times and the output is concatenated preserving the temporal dimension. The dense/LSTM block contains up to 4 layers with the same number of units, ranging from 20 to 100 in intervals of 5, and for a dense block the same layers are applied to all lead times like the convolutional layers. These fully connected layers are more prone to overfitting, so ℓ_2-regularization was applied to their weights. In the case dense layers are used, there is an extra unregularized dense layer with 20 units applied to all lead times. All of these layers use the ReLU activation function.

The "peephole" connection feeds the HA40 wind speed forecast central grid point and its standard deviation over the grid to a concatenation layer that aggregates with the output of the previous dense/LSTM block and the seecondary local input. The concatenated data is then passed to an output dense/LSTM layer that outputs the values of μ_t, σ_t. This output layer uses the softplus activation to ensure strictly positive output. The peephole allows the network to learn simple linear relationships as discussed in the section about the Wide & Deep architecture.

In the case no LSTM layers are used, only dense ones, the ouput dense layer receives the flattened array obtained from concatenation, and it outputs $49 * 2 = 98$ values corresponding to the μ_t, σ_t. This way the final output layer is responsible for aggregating the temporal information. Finally, networks for ARMOS(p) forecasts require estimating the extra ϕ_j parameters, which are global (no temporal dependence) and therefore have no benefit over the LSTM architecture. For this reason, they follow an alternative shallow dense path, which is composed of two dense layers with p neurons. The first is applied to all lead times and uses ReLU activations, with output of dimension $(49, p)$. This output is flattened and used as input by the final layer to calculate the value of the ϕ_j. It is expected that $\phi_j > 0$ since we know that the correlation between lead times is positive, and it is required that $\phi_j < 1$ to maintain the stability of the AR process, so we use sigmoid activation functions to ensure the output is in the correct domain.

In total we compare 8 different network models. The EMOSnet/SS uses dense hidden and output layers while the LSTM/EMOSnet/SS applies the LSTM layers. In practice the LSTM substantially improves the results on the validation set so we have used it for the ARMOS(p)net models $p = 1, 2, 3$, which have the same architecture as the LSTM/EMOSnet/SS but have extra dense layers for estimation of the autoregressive parameters. Finally, we also consider the ARMOS(p)net/SS to verify if the Schaake Shuffle improves multivariate forecast quality.

The hyperparameters optimized for the neural networks are the input grid size, number of convolutional layers per filter size, number of dense/LSTM layers, number of neurons per layer, and the strength of ℓ_2-regularization. We have selected the optimal configuration via Bayesian Optimization with around 80 trials per model.

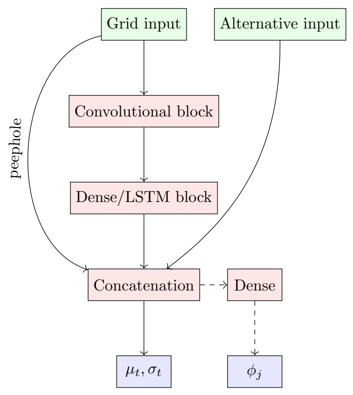
During model selection all models were trained on the training split of the training dataset and the ones with best performance on the validation split (unseen data) were selected as best models. After that, the selected models were trained on the full training dataset for verification in the test dataset, which was completely left out of all parts of the selection and training procedures. In order to stabilize the training of the models, all variables in the training dataset except HA40 wind speed forecasts and observations were normalized.

All models and automated tuning procedures were implemented in Python using TensorFlow/Keras [1, 9], the Tensorflow Probaility extension and the Keras Tuner package [32]. Tuning and training of the network models was performed at the Snellius Dutch national supercomputer on a GPU computing node.

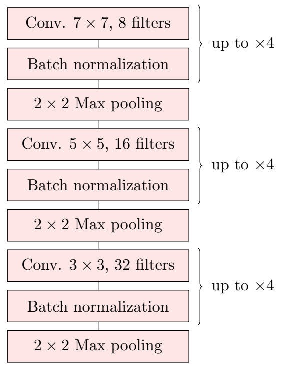
5.3 Results

The selected global linear EMOS and ARMOS calculated the variance over an 11×11 grid (size 5). Forward selection showed that the inclusion of all variables had a positive impact on the validation loss, so the final model used all of the 4 auxiliary variables. The local selection showed very similar results, for most stations the optimal grid size and variable selection had the similar results as for the global models, so the 11×11 grid and all 4 variables were used in all local and global EMOS/ARMOS models. The results of hyperparameter tuning for the neural networks are displayed in Table 3.

After training the models on the full training dataset, we computed the average scores over the test dataset. The results are displayed in Table 4. The LS for the ARMOS(p)(net)/SS models is



(a) Structural diagram.



(b) Convolutional block

Figure 13: General neural network architecture for estimation of EMOS and ARMOS models.

	EMOSnet	LSTM/EMOSnet	ARMOS(1)net	ARMOS(2)net	ARMOS(3)net
Grid size	10	50	50	30	30
7×7 conv.	4	4	4	1	1
5×5 conv.	4	0	0	2	3
3×3 conv.	4	1	3	1	0
Dense/LSTM	4	0	0	3	1
Units	60	20	20	60	40
ℓ_2-reg.	0.0005	0.0001	0.0002	0.0012	0.0004

Table 3: Selected hyperparameters for each network. "Grid size" is the size of the input grid of wind speed forecasts, the "$n \times n$ conv." refers to the number of convolutional layers with kernels of size $n \times n$ in the convolutional block, "dense/LSTM" refers to the number of layers, "units" to the number of neurons in each dense/LSTM layer, and "ℓ_2-reg" is the strength of ℓ_2 regularization on the dense/LSTM layers.

omitted since it is equal to the LS when not computed from samples, so it is unchanged by the application of the Schaake Shuffle. It is clear that local models outperform their global counterparts and network models outperform both. A striking feature of the results is that ARMOS models outperform their EMOS counterparts with respect to the LS in every group, but underperform with respect to the ES. This discrepancy is probably explained by the fact that the ES is mostly insensitive to the correlation structure, focusing purely on marginal calibration, while the LS incorporates both aspects simultaneously. In Section 3 we observed how the marginal variance of ARMOS models introduces uncertainty from previous lead times, which could lead to less confident marginal forecasts when compared to EMOS.

A comparison of the score results between ARMOS models on all groups indicates that $p = 2$ is the optimal value. Also, the application of the Schaake Shuffle on ARMOS(1) and ARMOS(2) samples does seem to improve the VS of both local and global models at a small cost of the ES for global models, while it clearly deteriorates performance for the networks. The best model for the LS and VS is the ARMOS(2)net, while with respect to the ES the LSTM/EMOSnet/SS wins over the ARMOS(2)net by a slight margin.

Next we analyse the marginal verification results for the 36h and 48h lead times. These are considered the most interesting lead times because they are far from the initialization time, making forecasts for these lead times harder, and they present distinct behaviour with respect to the daily cycle. For completeness we also present marginal verification results for the 12h and 24h lead times in the Appendix. Figures 14 - 16 show the PIT plots, Brier Skill Scores and Reliability Diagrams for thresholds 5.0 m/s, 10.0 m/s and 15.0 m/s for the marginal distributions at the lead times mentioned above. Notice that applying the Schaake Shuffle does not alter marginal sampling, so we do not mention SS models.

In most plots the linear ARMOS(3) models showed anomalous behaviour, which further supports that the choice $p = 3$ is suboptimal, thus we leave it out of further discussion. Analysing the PIT

Global	LS	ES	VS
EMOS/SS	82.73	6.23	17.52
ARMOS(1)	77.06	7.08	23.14
ARMOS(2)	72.07	7.28	19.43
ARMOS(3)	72.40	9.87	18.324
ARMOS(1)/SS	-	6.80	18.89
ARMOS(2)/SS	-	7.03	17.41
ARMOS(3)/SS	-	8.95	21.26
Local	**LS**	**ES**	**VS**
EMOS/SS	78.10	6.19	17.30
ARMOS(1)	73.81	6.40	21.92
ARMOS(2)	70.03	6.92	19.06
ARMOS(3)	70.19	7.88	18.08
ARMOS(1)/SS	-	6.60	19.03
ARMOS(2)/SS	-	7.12	17.65
ARMOS(3)/SS	-	8.09	19.11
Network	**LS**	**ES**	**VS**
EMOSnet/SS	80.05	6.10	17.86
LSTM/EMOSnet/SS	72.60	**5.61**	16.52
ARMOS(1)net	61.17	5.75	16.24
ARMOS(2)net	**60.54**	5.73	**16.08**
ARMOS(3)net	61.05	5.74	16.48
ARMOS(1)net/SS	-	5.84	17.53
ARMOS(2)net/SS	-	5.80	17.68
ARMOS(3)net/SS	-	5.85	17.42

Table 4: Average scores on test dataset.

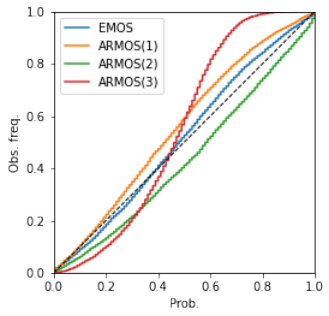
diagrams for the 36h and 48h lead times we observe that the linear EMOS and ARMOS(1) models show good calibration (Figures 14a - 14d), with slight positive bias for the global ARMOS(1) model at 36h (Figure 14a), while the linear ARMOS(2) models tends to have a negative bias. The best calibrated network for those lead times is clearly the LSTM/EMOSnet, while the ARMOSnets tend to be overconfident and the EMOSnet has a negative bias at the 36h lead time and is underconfident at the 48h lead time (Figures 14e - 14f). The results are similar for the 12h and 24h lead times since they match the 36h and 48h lead times in the daily cycle. One key difference is that the EMOSnet outperforms the LSTM/EMOSnet in the 12h and 24h lead times (Figures 17e and 17f of the Appendix).

The Brier Skill Score indicates the performance of the models with respect to climatology (average number of positive outcomes on the training dataset for each lead time and threshold), and the BSS of global linear EMOS is presented as benchmark in all graphs. The BSS indicates that in general the performance of the linear EMOS models and their linear ARMOS(1) counterparts are somewhat equivalent for the 36h and 48h lead times (Figures 15a - 15d), with local models outperforming global ones and the local ARMOS(1) being slightly worse than local EMOS at the 36h lead time for all thresholds (Figure 15c). The linear ARMOS(2) model generally underperforms when compared to the other two linear models, with the exception of the global ARMOS(2) for the 12h lead time (Figure 18a of the Appendix). At the 36h lead time (Figures 15e) we see that all networks outperform the global EMOS benchmark until around the 11.0 m/s threshold, at which all of them start to underperform with the exception of the LSTM/EMOSnet which is better for all thresholds up to the 16.0 m/s threshold. The same is true for the 12h lead time, but the networks performance is slightly worse in comparison to the benchmark. At the 48h lead time (Figure 15f) all networks perform better than Global EMOS for practically all lead times, with the LSTM/EMOSnet and ARMOS(2)net performing better for thresholds between 13.0 m/s and 18 m/s. In the 24h lead time (Figure 18f of the Appendix) the situation is similar, but the ARMOS(2) performs a little worse for thresholds higher than 13.0 m/s and the EMOSnet is much better, matching the LSTM/EMOSnet.

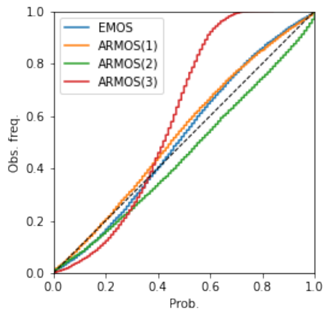
Finally, we analyse the Reliability diagrams. The linear EMOS and linear ARMOS(1) models are by far the most reliable linear models (Figures 16a - 16q) with their graph practically matching the main diagonal with the exception of the global ARMOS(1) (Figure 16a) which is slightly underconfident and for the 48h lead time and 15 m/s threshold (Figures 16p and 16q), for which all models show erratic behaviour. Notice that for the 15 m/s threshold the forecast probabilities concentrate heavily on lower values as seen in the frequency histogram, which explains why these reliability curves are not very representative. While linear EMOS and linear ARMOS(1) appear to be somewhat equivalent, the linear ARMOS(2) models show a negative bias that increases with the wind speed threshold. The linear models behave very similarly in terms of the probability distribution for each lead times and wind speed threshold.

The Reliability diagrams for the network models (Figures 16c - 16r), show that the LSTM/EMOSnet is the best calibrated network overall, while the ARMOSnet models are well calibrated for the 5.0 m/s threshold (specially for the 36h lead time) and increasing negative bias for higher thresholds. Although the 48h lead time for 15.0 m/s threshold also shows poor calibration for the (Figure 16r) with big negative bias for the ARMOSnets, there is a clear improvement when compared to the linear models (Figures 16p and 16q). In the case of linear models ARMOS(1) clearly outperformed ARMOS(2), but this is not the case for the network models. The ARMOS(2)net performs just as well as the ARMOS(1)net for the 5.0 m/s and 10.0 m/s thresholds (Figures 16c, 16f, 16l and 16o), both having a negative bias on the 36h lead time for the 10.0 m/s threshold (Figure 16i), but it is surprisingly better for the 15.0 m/s threshold (Figures 16i and 16r). The network models show big improvements for the ARMOS(2) model and specially for the ARMOS(3) model, they also show great improvement in discrimination as seen in the frequency histograms. The Reliability diagrams of the for the 12h and 24h lead times (Figures 19a - 19r of the Appendix) show somewhat the same behaviour for linear and network models.

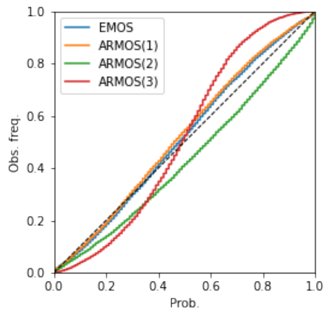
In general, EMOS and ARMOS(1) (both linear and network) models tend to have best performance with respect to marginal verification, with the LSTM/EMOSnet appearing to be the best overall. This is consistent with the fact that the LSTM/EMOSnet is the best model with respect to the Energy Score, a score that mainly penalizes for marginal calibration. Surprisingly however, the ARMOS(2)net also performs well with respect to the BSS and reliability diagrams, while its linear counterpart performs very poorly.



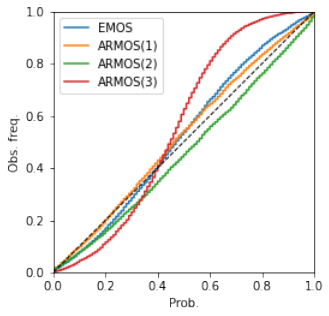
(a) Global 36h



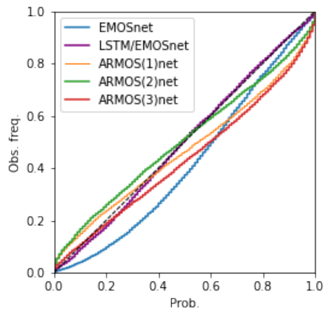
(b) Global 48h



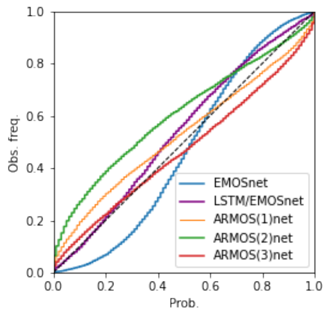
(c) Local 36h



(d) Local 48h

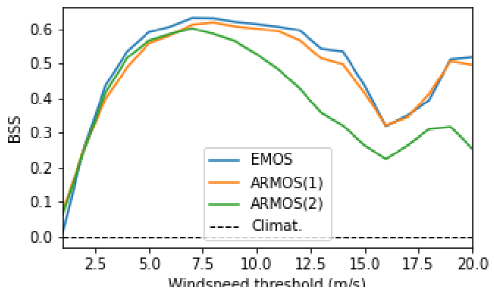


(e) Net. 36h

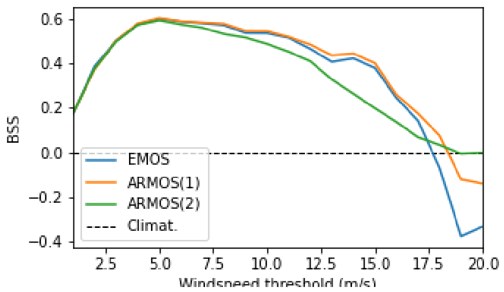


(f) Net. 48h

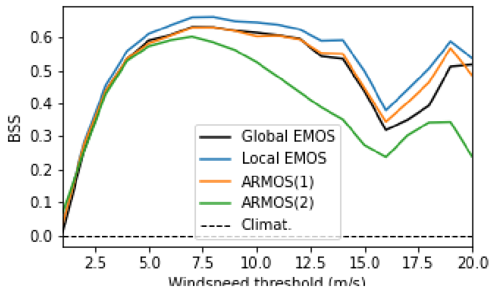
Figure 14: PIT diagrams for 36h and 48 lead times.



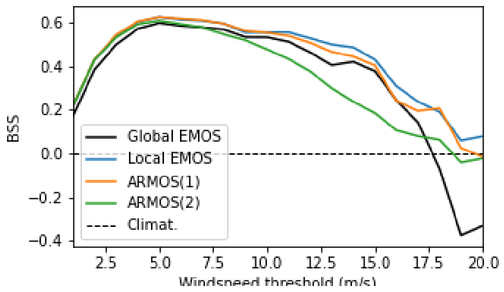
(a) Global 36h



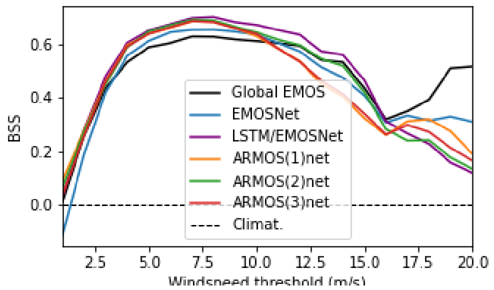
(b) Global 48h



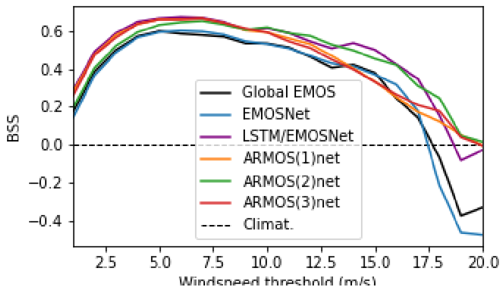
(c) Local 36h



(d) Local 48h

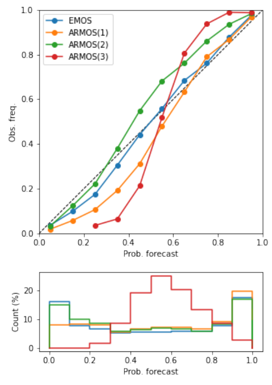


(e) Net. 36h

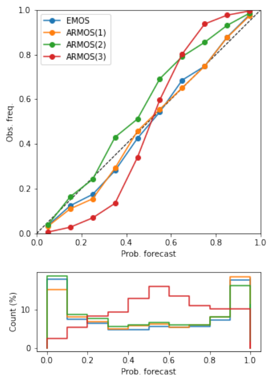


(f) Net. 48h

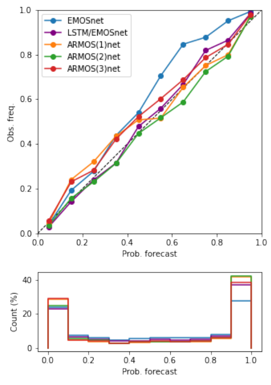
Figure 15: Brier Skill Scores relative to Global EMOS for 36h and 48h lead times. Global and local ARMOS(3) didn't fit in scale due to very poor performance.



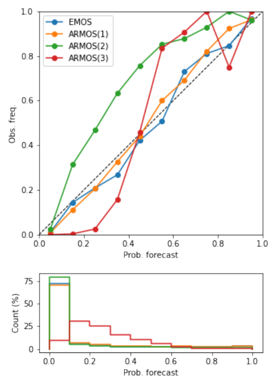
(a) Global 36h, 5 m/s



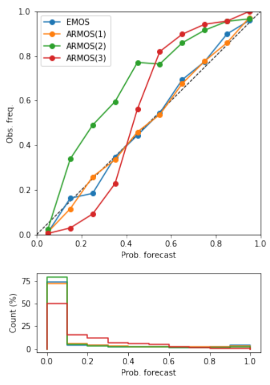
(b) Local 36h, 5 m/s



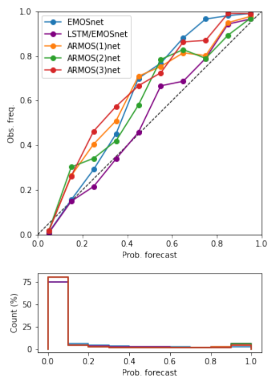
(c) Net. 36h, 5 m/s



(d) Global 36h, 10 m/s

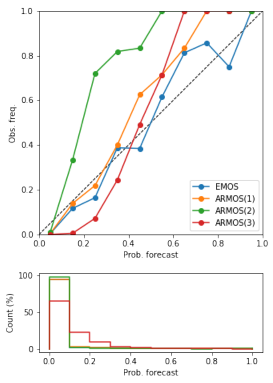


(e) Local 36h, 10 m/s

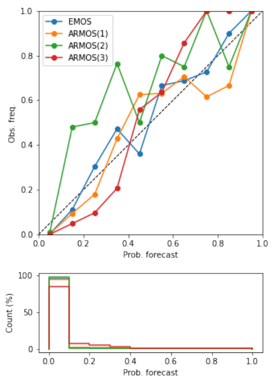


(f) Net. 36h, 10 m/s

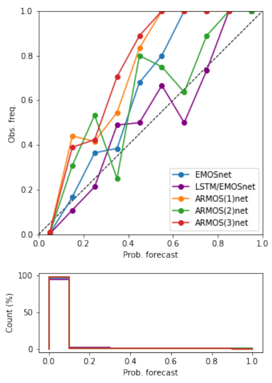
Figure 16: Reliability diagrams for lead times 36h and 48h and thresholds 5.0, 10.0 and 15.0 m/s.



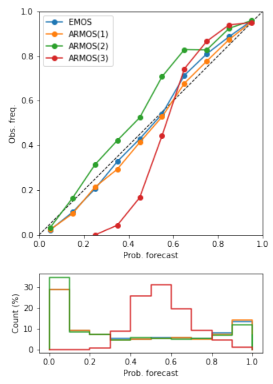
(g) Global 36h, 15 m/s



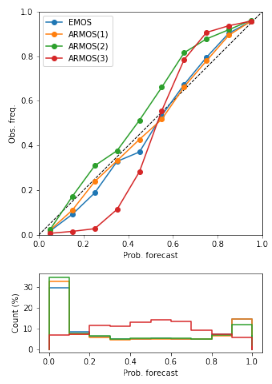
(h) Local 36h, 15 m/s



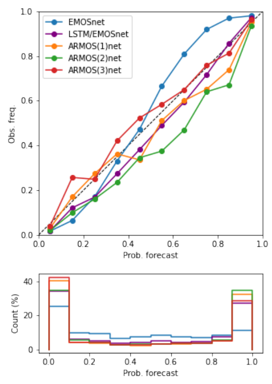
(i) Net. 36h, 15 m/s



(j) Global 48h, 5 m/s

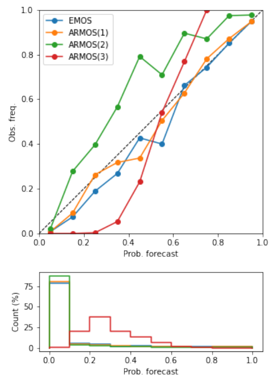


(k) Local 48h, 5 m/s

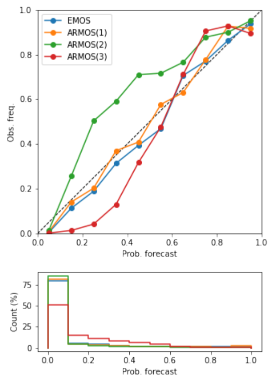


(l) Net. 48h, 5 m/s

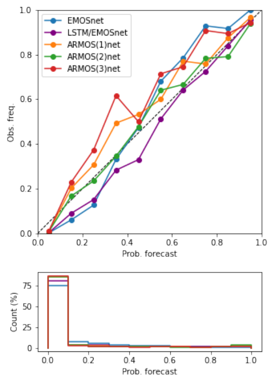
Figure 16: Reliability diagrams for lead times 36h and 48h and thresholds 5.0, 10.0 and 15.0 m/s.



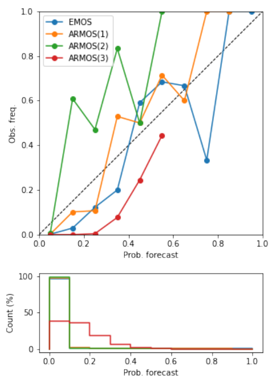
(m) Global 48h, 10 m/s



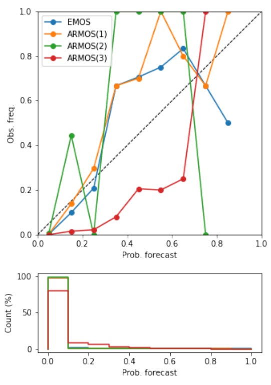
(n) Local 48h, 10 m/s



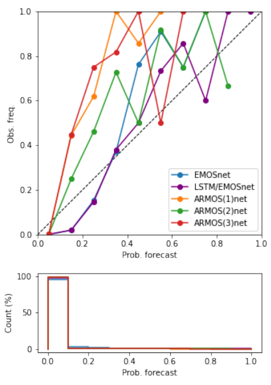
(o) Net. 48h, 10 m/s



(p) Global 48h, 15 m/s



(q) Local 48h, 15 m/s



(r) Net. 48h, 15 m/s

Figure 16: Reliability diagrams for lead times 36h and 48h and thresholds 5.0, 10.0 and 15.0 m/s.

6 Conclusion & Discussion

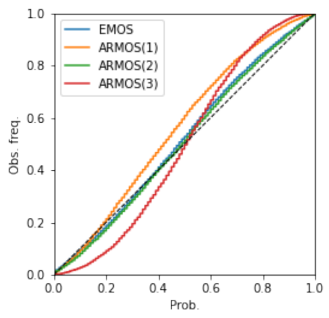
In this study we have compared two approaches to the multivariate statistical postprocessing of wind speed forecasts from the NWP Harmonie-Arome model from lead times 0h to 48h: the EMOS coupled with the Schaake Shuffle and the novel ARMOS(p) model, both using Truncated Normal distributions. For each of them we have further analysed different parameter estimation methods that incorporate spatio-temporal information in different ways. In total we have studied three groups of models: the global linear models, local linear models and neural network models. The linear models are distinguished by whether they have been estimated for all station simultaneously (global) or if an individual model is fitted for each stations independently (local). Linear models incorporate spatial information via the *variance* of the Harmonie-Arome wind speed forecasts in a neighbourhood of the station. On the other hand, neural network models use convolutional layers to extract additional spatial information from the grid and the LSTM to learn temporal dependencies.

The results show that neural networks perform very well compared to the linear models. Specially the ones that introduce the LSTM architecture for predictions of the mean and variance parameters. Furthermore, the ARMOS(p) models show improvements in performance with respect to the Logarithmic and Variogram Scores when compared to the EMOS/SS counterpart in each group, which indicates that the method is indeed better at modelling the lead time dependencies. However, EMOS/SS models tend to perform better with respect to the Energy Score. The marginal verification results agree with the ES results and show that EMOS/SS models have better calibrated marginal distributions, specially for predicting higher wind speeds. This makes sense since EMOS/SS models fit the marginal distributions directly, while the ARMOS(p) models fit the entire multivariate distribution at once. Additionally, marginal distributions of ARMOS(p) models inherit uncertainty from previous lead times, naturally making the marginal forecasts less confident. Overall the best models were the ARMOS(2)net and the LSTM/EMOSnet/SS. The first is by far the best with respect to the scores that evaluate the quality of the correlation structure, and the second is better at the marginal level.

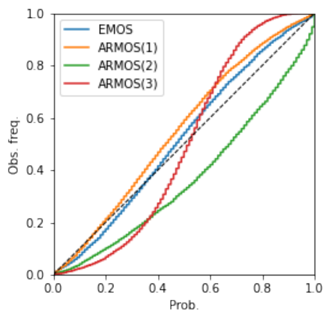
It is important to mention that during marginal validation the linear EMOS models have an advantage over the others. While ARMOS models and network models must share information through lead times during estimation, either via explicit temporal modelling or via parameter sharing, the linear EMOS models are fit independently for each lead time. Therefore they target optimization of marginal forecasts directly, while the others are forced to do it indirectly via global model optimization (global with respect to the time dimension). Besides, a possible solution to improve forecast of higher wind speeds would be to resample extreme wind speed occurrences during network training, or set higher weights to the gradients of those samples.

In further explorations, several improvements on network architecture could be considered, such as the application of residual convolutional layers [19], which could lead to much deeper network architectures without overfitting, or convolutional LSTM layers [39], which mix the convolutional and LSTM architectures. Also, it would be interesting to experiment with different marginal distributions other than the Truncated Normal such as the Log Normal or Weibull, since their fatter tail could improve forecasting of higher wind speeds.

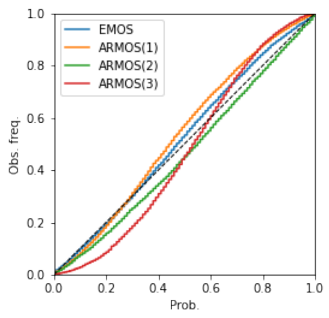
A Appendix



(a) Global 12h

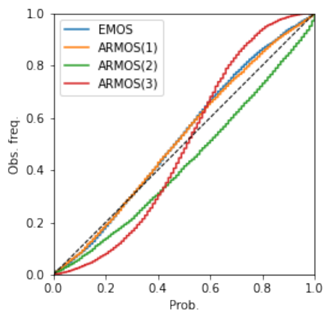


(b) Global 24h

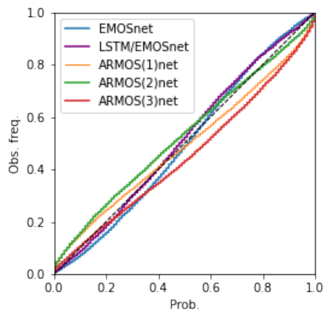


(c) Local 12h

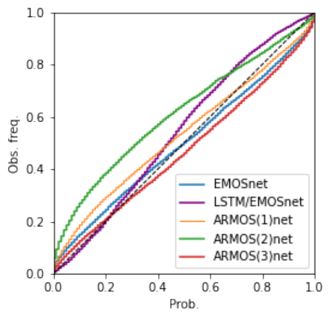
5



(d) Local 24h

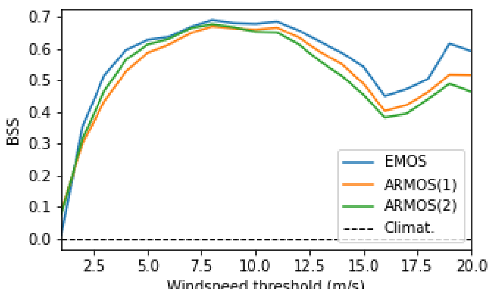


(e) Net. 12h

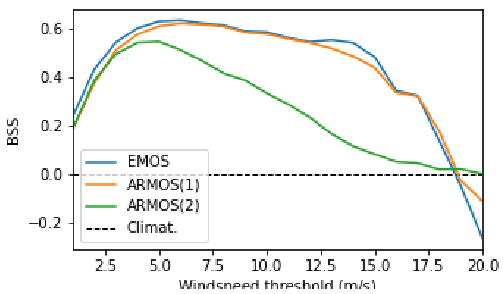


(f) Net. 24h

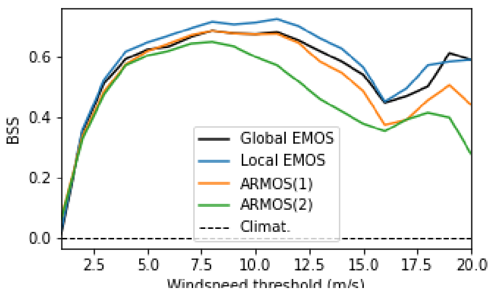
Figure 17: PIT diagrams for 12h and 24h lead times.



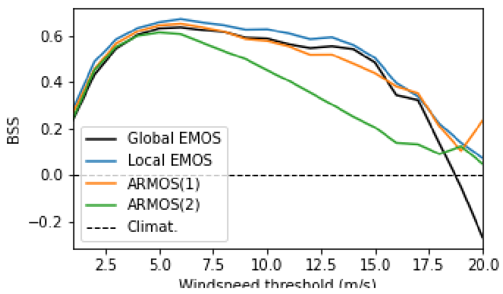
(a) Global 12h



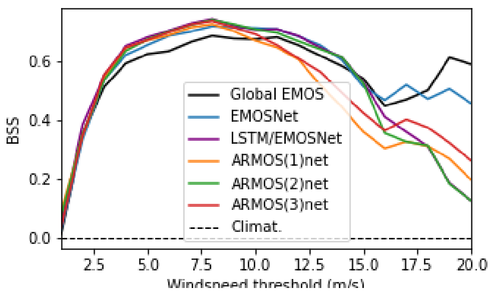
(b) Global 24h



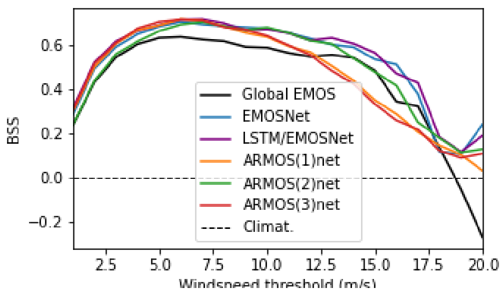
(c) Local 12h



(d) Local 24h

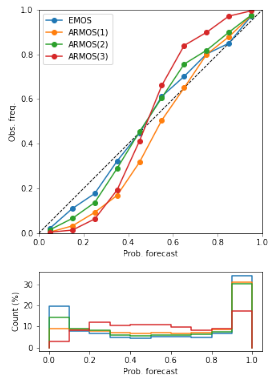


(e) Net. 12h

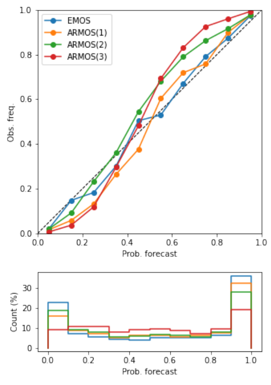


(f) Net. 24h

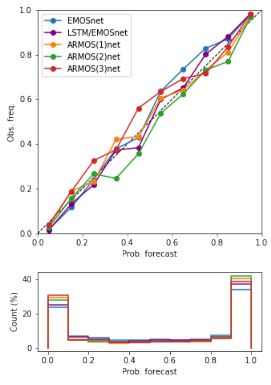
Figure 18: Brier Skill Scores relative to Global EMOS for 12h and 24h lead times. Global and local ARMOS(3) didn't fit in scale due to extremely poor performance.



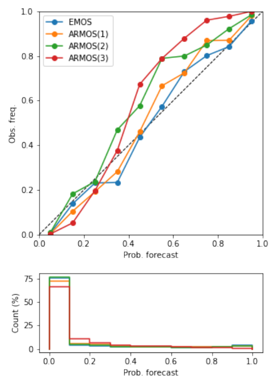
(a) Global 12h, 5 m/s



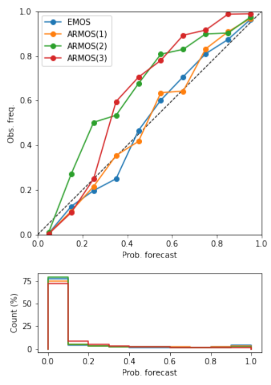
(b) Local 12h, 5 m/s



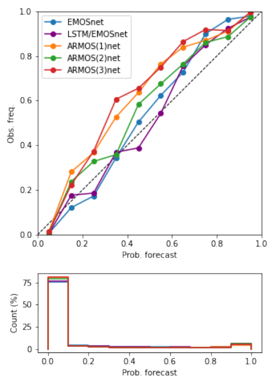
(c) Net. 12h, 5 m/s



(d) Global 12h, 10 m/s

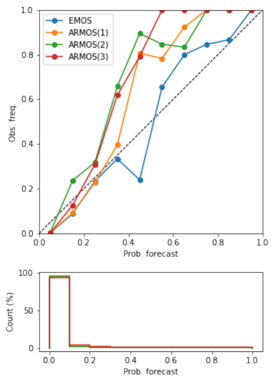


(e) Local 12h, 10 m/s

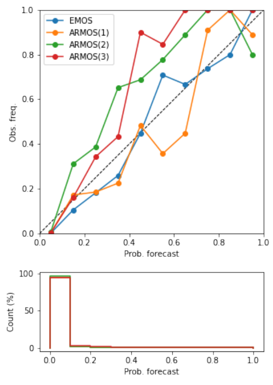


(f) Net. 12h, 10 m/s

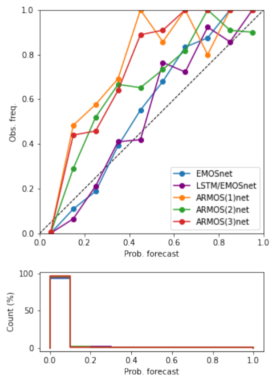
Figure 19: Reliability diagrams for lead times 12h and 24h and thresholds 5.0 10.0 and 15.0 m/s.



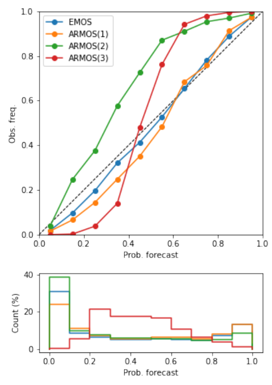
(g) Global 12h, 15 m/s



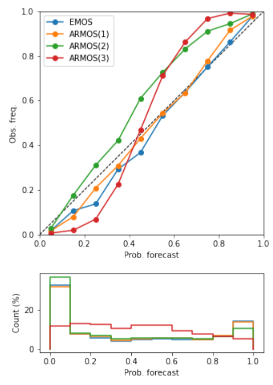
(h) Local 12h, 15 m/s



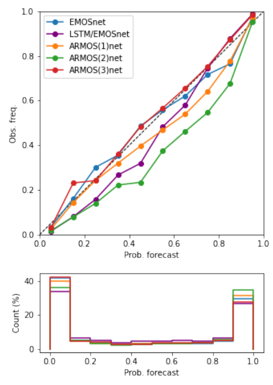
(i) Net. 12h, 15 m/s



(j) Global 24h, 5 m/s

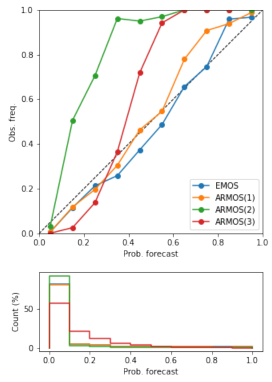


(k) Local 24h, 5 m/s

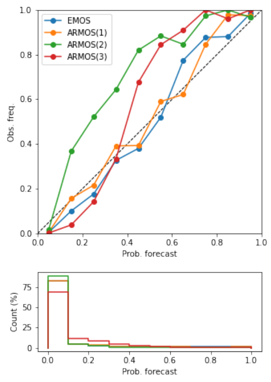


(l) Net. 24h, 5 m/s

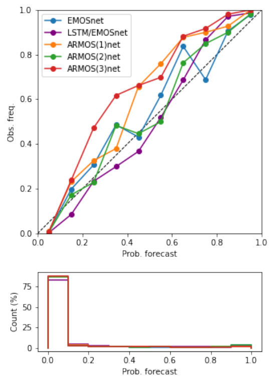
Figure 19: Reliability diagrams for lead times 12h and 24h and thresholds 5.0 10.0 and 15.0 m/s.



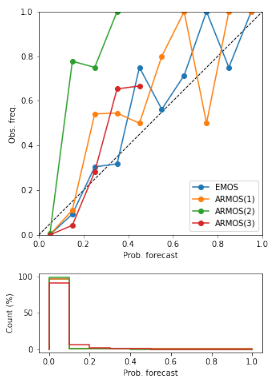
(m) Global 24h, 10 m/s



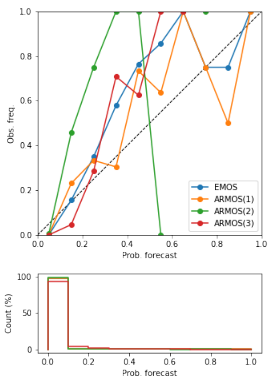
(n) Local 24h, 10 m/s



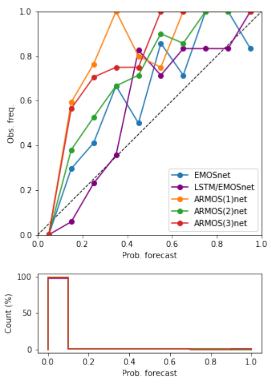
(o) Net. 24h, 10 m/s



(p) Global 24h, 15 m/s



(q) Local 24h, 15 m/s



(r) Net. 24h, 15 m/s

Figure 19: Reliability diagrams for lead times 12h and 24h and thresholds 5.0 10.0 and 15.0 m/s.

Location	Code	Lon (E^o)	Lat (N^o)
IJMOND	209	4.518	52.464
VOORSCHOTEN	215	4.436	52.140
IJMUIDEN	225	4.555	52.462
TEXELHORS	229	4.713	52.998
DE KOOY	235	4.781	52.927
SCHIPHOL	240	4.790	52.317
VLIELAND	242	4.917	53.242
WIJDENES	248	5.174	52.633
BERKHOUT	249	4.979	52.643
HOORN (TERSCHELLING)	251	5.346	53.391
HOUTRIBDIJK	258	5.401	52.648
DE BILT	260	5.180	52.099
STAVOREN	267	5.383	52.897
LELYSTAD	269	5.521	52.458
LEEUWARDEN	270	5.752	53.223
MARKNESSE	273	5.888	52.702
DEELEN	275	5.872	52.055
LAUWERSOOG	277	6.199	53.412
HEINO	278	6.259	52.434
HOOGEVEEN	279	6.573	52.749
EELDE	280	6.585	53.124
HUPSEL	283	6.657	52.068
HUIBERTGAT	285	6.398	53.574
NIEUW BEERTA	286	7.149	53.194
TWENTHE	290	6.891	52.273
CADZAND	308	3.379	51.380
VLISSINGEN	310	3.596	51.441
OOSTERSCHELDE	312	3.622	51.767
VLAKTE V.D. RAAN	313	3.242	51.504
HANSWEERT	315	3.998	51.446
SCHAAR	316	3.694	51.656
WESTDORPE	319	3.861	51.225
WILHELMINADORP	323	3.894	51.530
STAVENISSE	324	4.006	51.596
HOEK VAN HOLLAND	330	4.122	51.991
THOLEN	331	4.192	51.479
WOENSDRECHT	340	4.342	51.448
R'DAM-GEULHAVEN	343	4.312	51.892
ROTTERDAM	344	4.435	51.956
CABAUW	348	4.926	51.969
GILZE-RIJEN	350	4.935	51.565
HERWIJNEN	356	5.145	51.858
EINDHOVEN	370	5.377	51.450
VOLKEL	375	5.707	51.659
ELL	377	5.763	51.197
MAASTRICHT	380	5.762	50.905
ARCEN	391	6.196	51.497

Table 5: Available stations

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] Rami Al-Hajj, Ali Assi, Mohamad Fouad, and Emad Mabrouk. A hybrid lstm-based genetic programming approach for short-term prediction of global solar radiation using weather data. *Processes*, 9(7), 2021.

[3] Sándor Baran and Sebastian Lerch. Log-normal distribution based ensemble model output statistics models for probabilistic wind-speed forecasting. *Quarterly Journal of the Royal Meteorological Society*, 141(691):2289–2299, 2015.

[4] Atilim Gunes Baydin, Barak A Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. Automatic differentiation in machine learning: a survey. *Journal of Marchine Learning Research*, 18:1–43, 2018.

[5] Lisa Bengtsson, Ulf Andrae, Trygve Aspelien, Yurii Batrak, Javier Calvo, Wim de Rooy, Emily Gleeson, Bent Hansen-Sass, Mariken Homleid, Mariano Hortal, Karl-Ivar Ivarsson, Geert Lenderink, Sami Niemelä, Kristian Pagh Nielsen, Jeanette Onvlee, Laura Rontu, Patrick Samuelsson, Daniel Santos Muñoz, Alvaro Subias, Sander Tijm, Velle Toll, Xiaohua Yang, and Morten Ødegaard Køltzow. The harmonie–arome model configuration in the aladin–hirlam nwp system. *Monthly Weather Review*, 145(5):1919 – 1935, 2017.

[6] Mathias Blicher Bjerregård, Jan Kloppenborg Møller, and Henrik Madsen. An introduction to multivariate probabilistic forecast evaluation. *Energy and AI*, 4:100058, 2021.

[7] Peter J Brockwell and Richard A Davis. *Introduction to time series and forecasting*. Springer, 2002.

[8] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, Rohan Anil, Zakaria Haque, Lichan Hong, Vihan Jain, Xiaobing Liu, and Hemal Shah. Wide & deep learning for recommender systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, DLRS 2016, page 7–10, New York, NY, USA, 2016. Association for Computing Machinery.

[9] Francois Chollet et al. Keras, 2015.

[10] Martyn Clark, Subhrendu Gangopadhyay, Lauren Hay, Balaji Rajagopalan, and Robert Wilby. The schaake shuffle: A method for reconstructing space–time variability in forecasted precipitation and temperature fields. *Journal of Hydrometeorology*, 5(1):243 – 262, 2004.

[11] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In Geoffrey Gordon, David Dunson, and Miroslav Dudík, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 315–323, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR.

[12] Tilmann Gneiting, Fadoua Balabdaoui, and Adrian E. Raftery. Probabilistic forecasts, calibration and sharpness. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 69(2):243–268, 2007.

[13] Tilmann Gneiting and Adrian E. Raftery. Weather forecasting with ensemble methods. *Science*, 310(5746):248–249, 2005.

[14] Tilmann Gneiting and Adrian E Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477):359–378, 2007.

[15] Tilmann Gneiting, Adrian E. Raftery, Anton H. Westveld, and Tom Goldman. Calibrated probabilistic forecasting using ensemble model output statistics and minimum crps estimation. *Monthly Weather Review*, 133(5):1098 – 1118, 2005.

[16] I. J. Good. Rational decisions. *Journal of the Royal Statistical Society: Series B (Methodological)*, 14(1):107–114, 1952.

[17] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[18] Thomas M. Hamill. Interpretation of rank histograms for verifying ensemble forecasts. *Monthly Weather Review*, 129(3):550 – 560, 2001.

[19] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, Los Alamitos, CA, USA, jun 2016. IEEE Computer Society.

[20] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[21] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

[22] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 448–456, Lille, France, 07–09 Jul 2015. PMLR.

[23] C. G. Justus, W. R. Hargraves, Amir Mikhail, and Denise Graber. Methods for estimating wind speed frequency distributions. *Journal of Applied Meteorology (1962-1982)*, 17(3):350–353, 1978.

[24] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[25] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.

[26] Sebastian Lerch and Thordis L. Thorarinsdottir. Comparison of non-homogeneous regression models for probabilistic wind speed forecasting. *Tellus A: Dynamic Meteorology and Oceanography*, 65(1):21206, 2013.

[27] Edward N. Lorenz. Deterministic nonperiodic flow. *Journal of Atmospheric Sciences*, 20(2):130 – 141, 1963.

[28] James E. Matheson and Robert L. Winkler. Scoring rules for continuous probability distributions. *Management Science*, 22(10):1087–1096, 1976.

[29] Jonas Mockus, Vytautas Tiesis, and Antanas Zilinskas. The application of bayesian methods for seeking the extremum. *Towards global optimization*, 2(117-129):2, 1978.

[30] Annette Möller and Jürgen Groß. Probabilistic temperature forecasting based on an ensemble autoregressive modification. *Quarterly Journal of the Royal Meteorological Society*, 142(696):1385–1394, 2016.

[31] Annette Möller and Jürgen Groß. Probabilistic temperature forecasting with a heteroscedastic autoregressive ensemble postprocessing model. *Quarterly Journal of the Royal Meteorological Society*, 146(726):211–224, 2020.

[32] Tom O'Malley, Elie Bursztein, James Long, François Chollet, Haifeng Jin, Luca Invernizzi, et al. Keras Tuner. https://github.com/keras-team/keras-tuner, 2019.

[33] Pierre Pinson and Julija Tastu. *Discrimination ability of the Energy score.* Number 15 in DTU Compute-Technical Report-2013. Technical University of Denmark, 2013.

[34] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[35] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[36] Roman Schefzik, Thordis L. Thorarinsdottir, and Tilmann Gneiting. Uncertainty Quantification in Complex Simulation Models Using Ensemble Copula Coupling. *Statistical Science*, 28(4):616 – 640, 2013.

[37] Michael Scheuerer and Thomas M. Hamill. Variogram-based proper scoring rules for probabilistic forecasts of multivariate quantities. *Monthly Weather Review*, 143(4):1321 – 1334, 2015.

[38] Sanjib Sharma, Ganesh Raj Ghimire, and Ridwan Siddique. Machine learning for postprocessing ensemble streamflow forecasts. *CoRR*, abs/2106.09547, 2021.

[39] Xingjian Shi, Zhourong Chen, Hao Wang, Dit Yan Yeung, Wai Kin Wong, and Wang Chun Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. *Advances in Neural Information Processing Systems*, 2015-January:802–810, 2015. 29th Annual Conference on Neural Information Processing Systems, NIPS 2015 ; Conference date: 07-12-2015 Through 12-12-2015.

[40] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[41] Simon Veldkamp, Kirien Whan, Sjoerd Dirksen, and Maurice Schmeits. Statistical postprocessing of wind speed forecasts using convolutional neural networks. *Monthly Weather Review*, 149(4):1141 – 1152, 2021.

[42] Daniel S Wilks. *Statistical methods in the atmospheric sciences*, volume 100. Academic press, 2011.

[43] Yong Yu, Xiaosheng Si, Changhua Hu, and Jianxun Zhang. A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures. *Neural Computation*, 31(7):1235–1270, 07 2019.