Utrecht University

Koninklijk Nederlands Meteorologisch Instituut
Ministerie van Infrastructuur en Waterstaat

Faculteit Bètawetenschappen

Improving Probabilistic Forecasting of Extreme Wind Speeds in the Netherlands

Master Thesis

Simon Hakvoort

Mathematical Sciences

Supervisors:

Dr. Sjoerd Dirksen
Utrecht University

Dr. Bastien François
KNMI

Dr. Kirien Whan
KNMI

Second Reader:

Dr. Chiheb Ben Hammouda
Utrecht University

October 31, 2024

Abstract

Forecasting wind speeds is important because of its large impact on society. The forecasts are issued by Numerical Weather Prediction (NWP) models. These NWP models often contain biases and have errors in dispersion, therefore they are subjugated to statistical post-processing. A commonly used method to perform post-processing is ensemble model output statistics (EMOS), where the goal is to fit the parameters of a probability distribution based on NWP output. In early versions of EMOS, linear regression was employed for this task. In recent approaches, more complex models such as neural networks have been introduced. While neural networks are able to significantly improve performance up to medium range wind speeds, they struggle with high wind speeds. The models are often trained using the continuous ranked probability score (CRPS), a proper scoring rule that equally weighs all possible forecast values. In this work, we propose using a weighted version of the CRPS (wCRPS) to address the challenges associated with extreme wind speeds. The wCRPS is a proper scoring rule that emphasizes particular regions of the forecast through a weight function. We also explore different parametric distributions, namely the truncated normal (TN), log-normal (LN), generalized extreme value (GEV) and mixture distributions. Our findings suggest that using the wCRPS with an appropriate weight function can enhance performance on extremes. However, for models using linear regression, we observed a body-tail trade-off, where increased performance on extremes came at the cost of worse predictions for average wind speeds. We developed an approach where the weight function is selected based on user preference by selecting hyperparameters using a multi-objective optimization algorithm. For the convolutional neural network-based models, we found that with an appropriate weight function the performance on extremes could be increased. Further investigation on the weight function of neural network-based models is advised, as the best choice of weight function may not have been included in our search space. Additionally, the best-performing weight function is shown to be model-specific. Regarding the choice of distribution, no significant effect was observed.

Acknowledgements

I want to start by expressing my thanks to my supervisors, Dr. Sjoerd Dirksen, Dr. Bastien François and Dr. Kirien Whan. Their guidance, support, and thoughtful feedback throughout this project have been invaluable. They created a positive and motivating environment, always encouraging me and helping me explore new ideas. I would also like to extend my deepest gratitude to my family and friends. Your support and encouragement have meant a lot to me, especially during the challenging moments of this journey.

Contents

1 Introduction

Forecasting weather is a crucial task with significant implications for society, impacting both industries and individuals who rely on accurate predictions for their planning and decision-making. The stakes are even higher when it comes to predicting extreme weather events, such as severe windstorms, which can have devastating effects. Given their potential to cause widespread damage, improving the prediction of these extremes is essential.

One of the most commonly used tools for weather forecasting is the *Numerical Weather Prediction* (NWP) model [1], such as the Harmonie-Arome model [2], used by the Royal Netherlands Meteorological Institute (KNMI). These models use computer simulations to predict future weather conditions based on current observations, achieved by dividing the atmosphere into a grid and numerically solving differential equations to simulate atmospheric changes. When only a single run of an NWP model is performed, the result is known as a *deterministic* forecast. However, deterministic forecasts come with limitations. Even a slight error in the initial conditions can rapidly amplify over time, leading to inaccurate predictions [3]. Additionally, deterministic forecasts do not account for the inherent uncertainty in weather systems.

To address these issues, forecasters often generate an *ensemble* of forecasts by running the NWP model multiple times with slightly varied initial conditions [4, 5]. This approach introduces some level of uncertainty quantification, but it still has its drawbacks. Ensemble forecasts can suffer from systematic biases and errors due to model limitations and inaccuracies in initial conditions [1].

To mitigate the limitations of ensemble forecasts and improve their accuracy, *post-processing* techniques are commonly employed [1]. Post-processing involves applying statistical methods to the raw outputs of NWP models to correct systematic biases, account for underdispersion, and generate calibrated probabilistic forecasts. One widely used post-processing approach is *Ensemble Model Output Statistics* (EMOS) [6]. The goal of using EMOS is to fit the parameters of some probability distribution based on the output of the NWP model. Traditionally, linear models are used to model the relations between the output of the NWP model and the parameters. However, recent approaches use more advanced models, such as neural networks [7–12]. These models provide more skillful forecasts compared to their linear counterparts. However, so far the increase in skill by using neural-network based models occurs mainly for low-to-medium range wind speeds. For extreme wind speeds, the methods are typically outperformed by linear EMOS approaches or even climatology. This is an important motivation for this work, to look into methods to increase the performance of neural network-based EMOS models for extreme wind speeds.

Improving predictions for extremes is a delicate task. When training a model to better predict the occurrences of extremes, a possibility that comes to mind is constructing a dataset that contains more extremes compared to the observed data. However, this is deemed unwise as is mentioned in [13]. In essence, when forecast evaluation is based on the occurrence of an extreme event, consistently predicting extreme events becomes an appealing strategy. With media attention often focused on extreme occurrences, accurate forecasts may still be perceived as failures by the public, in case a model was unable to predict an extreme. This can lead to the temptation to rely on flawed decision-making processes. This phenomena is known as the *forecaster's dilemma*. This could be the result of training a model on a dataset that is not representative of real data, because it only contains extremes. We need to be aware of this dilemma, to ensure that our forecasts are not only accurate for extreme events but also reliable for the entirety of the data that is available.

In this work we will focus on improving the performance on extremes for EMOS with both linear and neural network-based models. We will conduct our investigation for deterministic Harmonie-Arome wind speed forecasts with a lead time of 48 hours. There are two techniques that we are investigating, the first one being the choice of parametric distribution in EMOS. The distributions which are considered are the truncated normal (TN), log-normal (LN), generalized extreme value (GEV), and mixtures of these distributions. The second way we will look into improving predictions of extremes is with weighted scoring rules. Traditionally, the *continuous ranked probability score* (CRPS) and the log-score are used to train the models. We will use the weighted continuous ranked probability score (wCRPS) to train the models, which introduces a weight function into the CRPS to emphasize the importance of certain events. The weight functions considered in this work are vertically shifted Gaussian CDFs, where the parameters in the weight function are tuned using cross-validation. We will use a variety of verification measures, such as reliability and cPIT diagrams, as well as Brier skill scores.

For linear models, we observed that using the wCRPS as a loss function during training improved performance on extremes. However, this increase in performance for extremes came with a decrease in performance for low- to medium wind speeds, a *body-tail trade-off*. This was especially prevalent in the wCRPS for weight functions that focus only on extremes, such as a Heaviside function. This specific weight function in the wCRPS will be abbreviated by the threshold-weigthed continuous ranked probability score (twCRPS). We obtained improved results using a Gaussian CDF with an additional vertical shift as weight function in the wCRPS. This significantly reduces the worsening in performance for lower wind speeds, while still increasing performance on extremes. For the neural networks, we used a convolutional neural network (CNN) architecture. The performance of CNNs varied a lot because of random initialization and stochastic gradient descent; hence we chose to use a bagging approach, where we combined the predictive distributions of multiple models. These models were much more consistent compared to single CNNs. With neural networks-based models, choosing weight functions that solely focus on extremes, like a Heaviside function, did not perform better on extremes, it only led to worse performance. Other weight functions were able to improve performance, such as a close to affine function with a positive slope. Using this affine function that was deemed optimal for the neural network based models did not lead to better performance on the linear performance, similar to how the best weight functions from the linear models could not improve performance for the neural network based models. This indicates that the optimal choice of weight function in the wCRPS is model-specific, and should be carefully studied before a choice is made. For the CNN-based models we did not observe the body-tail trade-off. Regarding the choice of distribution, we were unable to find a significant difference between the TN and the LN. The GEV distribution was more unstable than the TN and LN. Similarly, for the mixture and adaptive mixture distribution we did not observe an improvement compared to the TN distribution.

1.1 Related Work

There have been several contributions to improve the EMOS model for extremes. For example, [14] and [15] considered the LN and GEV as forecasting distributions and found that these improved over the TN distribution in terms of twCRPS with high thresholds. To avoid forecasting negative wind speeds when using the GEV distribution, the authors of [16] propose a truncated GEV distribution.

Several authors already proposed to use a loss function to focus on extremes, such as in [17, 18]. These studies use neural networks to generate point forecasts, where the forecast contains only a single number. Both find that using a weighted loss function leads to improved performance of the network on extremes, compared to non-weighted counterparts. However, in [17] the authors observe that this improvement in performance does come with increased overcasting and false alarm ratio.

We will use a similar strategy with choosing a weighted loss function. However, this will be done in combination with probabilistic forecasting, where the resulting forecast is a probability distribution. Independently of our work, in [19] an analogous strategy is used. The authors study 10 meter wind speed with an 18-member ensemble forecast in the United Kingdom with a lead time of 48 hours. They train linear EMOS models with the twCRPS using the Broyden-Fletcher-Goldfarb-Shanno (BFGS) optimizer, falling back to the Nelder-Mead in case this fails. They test both the truncated normal and truncated logistic as parametric distribution. They show that using the twCRPS as loss function leads to an improvement in performance for extremes wind speeds, at the cost of worse performance for moderate wind speeds. The authors call this a body-tail trade-off. To address the body-tail trade-off, they propose two approaches: the first involves training on a linear combination of the CRPS and twCRPS, while the second approach uses a convex combination of models trained separately on the CRPS and twCRPS. On our data, we also observe the body-tail trade-off for the linear models. The scope of this research is broader, as we use the wCRPS as a loss function and examine its effect on EMOS with linear regression as well as neural networks. The effectiveness of different weight functions for the wCRPS is assessed, along with strategies for managing the body-tail trade-off. Additionally, the impact of the choice of loss function on the selection of the parametric distribution is analyzed.

This thesis is structured as follows: firstly, in section 2 we describe the used dataset. Next, in section 3, we discuss probabilistic forecasting, including various methods to evaluate performance. Following in section 4, it is explained how EMOS works and how it can be used with neural networks. Finally, we present the results of our experiments in section 5 and conclude with a discussion of our findings in section 6.

2 Data

We will be using data from the Harmonie-Arome model cycle 40 (HA40) [2], which is a deterministic NWP forecast. This contains a 300×300 discretized grid, with spacings of 2.5 kilometers covering the Netherlands. We will be working with the set of forecasts initialized at 0000 UTC with a lead time of 48 hours.

The forecasts contain numerous predicted weather variables. The predictors that are used in this project are shown in Table 1. This set of predictors is the same as in [8]. In [8] the predictors were chosen using forward selection.

Predictors
Wind speed at 10 meter height
Mean sea level pressure
Turbulent kinetic energy
Humidity at 2 meter height
Geopotential height at 500 hPa

Table 1: Predictors used from the HA40 forecast

The predictand data will consist of 10-minute average wind speed observations, measured at a height of 10 meters, from 47 different Dutch weather stations. The grid point in which the weather station is located will be used to extract the predictor data. In case a grid of wind speeds is used, we will have a grid with an odd side length such that the weather station is located in the center of the grid. In Figure 1 the locations of the weather stations are displayed.

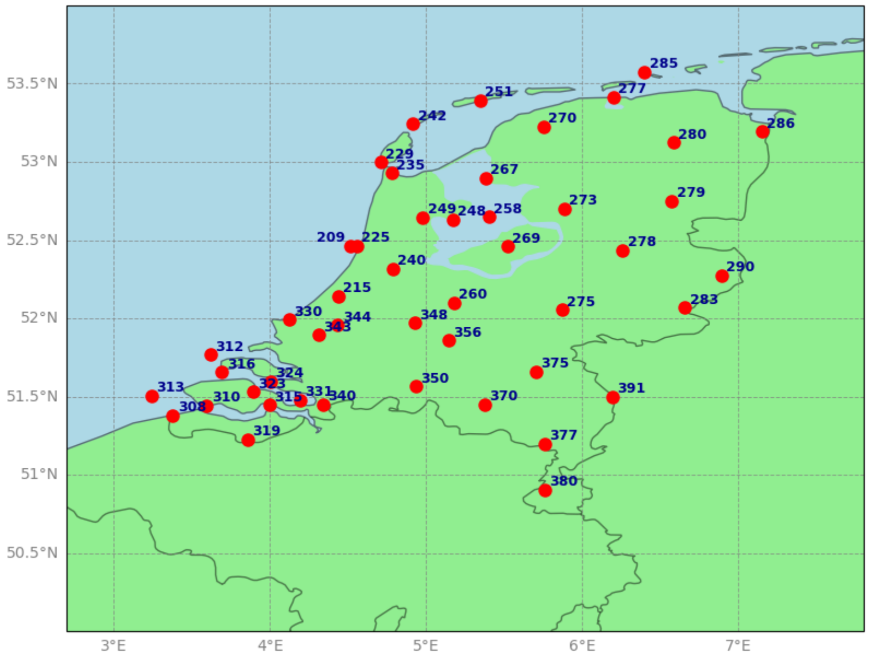


Figure 1: Locations of the Dutch weather station, including their unique station codes. Table 8 in Appendix B contains more information regarding the stations.

The data is separated into different folds for model selection using cross-validation, with a separate test set reserved for evaluating model performance.

Fold 1	October - December 2015 and January - March 2016
Fold 2	October - December 2016 and January - March 2017
Fold 3	October - November 2017 and January - March 2015
Test Set	November - December 2018, January - March 2019 and October - November 2019

Table 2: Folds 1, 2 and 3 will be used for model selection and validation, the test set is then used to evaluate the performance of the models.

The folds are split in this particular way to minimize the effect of autocorrelation between different folds. For cross-validation, the model is trained on two of the folds and validated on the remaining one. This process is repeated three times, each time with a different fold used for validation and the other two for training. The model's performance is then averaged over the three iterations to provide an overall performance estimate. Once we have selected the best hyperparameters using cross-validation, we will train the model using these selected hyperparameters on all three folds and evaluate the performance of the model on the test set.

Table 3 contains some statistics from the observations for the training data set. We can see that the first fold contains the most extreme observations, while the second fold was a relatively calm winter period. All folds are approximately the same size.

	Size	Mean (m/s)	90 percentile (m/s)	95 percentile (m/s)	99 percentile (m/s)
Training Data	23513	5.16	9.87	11.88	15.43
Fold 1	7754	5.62	10.84	12.91	16.32
Fold 2	7979	4.52	8.41	10.02	13.26
Fold 3	7780	5.37	10.29	12.21	15.44

Table 3: Statistics for the different folds. The row 'Training Data' consists of all three folds combined.

We used all the observations from the stations that fall in the range of dates from Table 2, except for the data from stations 229, 285 and 323 (Texelhors, Huibertgat and Wilhelminadorp). For these stations there was a significant amount of missing data; therefore we omitted all information regarding these stations.

In Figure 2 we can see a histogram with the observed wind speeds from the training set. Here we can clearly see how the data is distributed, the bulk of the observations is between 0 and 10 m/s, while there is a long tail for the higher wind speeds.

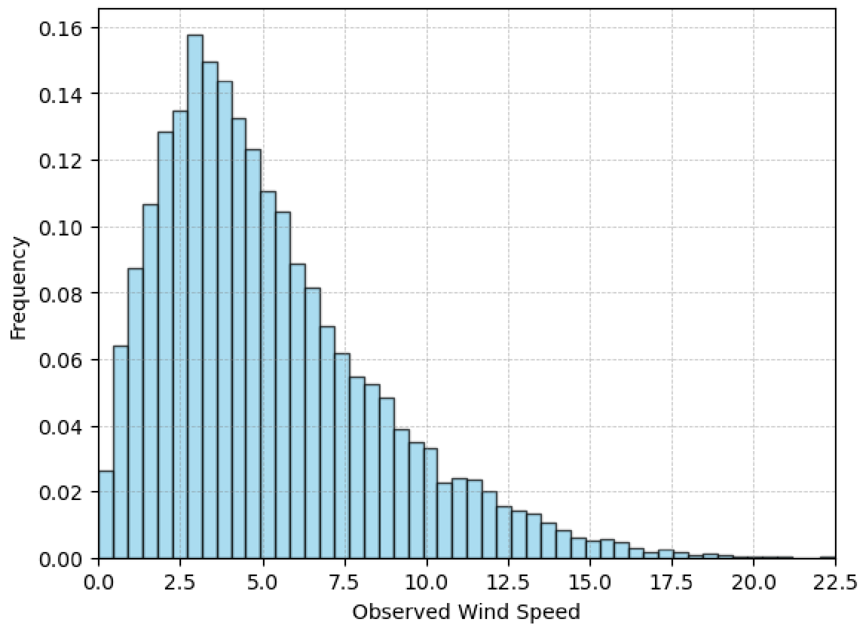


Figure 2: Frequency histogram of the observed wind speeds in the full training data set.

3 Probabilistic Forecasting

In this section, we provide an overview of probabilistic forecasting. First, we will discuss the advantages of probabilistic forecasts over deterministic forecasts. After that, an explanation of proper scoring rules is given, followed by weighted scoring rules. Then the different applied verification measures are further explained.

3.1 Deterministic vs Probabilistic

There are two types of forecasts we consider. The first is a deterministic forecast, which is also the output of the HA40 model. This forecast gives a single number as forecast. A significant drawback of deterministic forecasts is their inability to show uncertainty [20]. For example, in a deterministic forecast which states that the predicted wind speed is 10 meters per second (m/s), it is impossible to know how likely it is that the wind speed will be above 11 m/s. This makes decision-making based on a deterministic forecast a difficult task. For this reason, probabilistic forecasting has come into play. With probabilistic forecasting, the output is a probability distribution in which we can capture the uncertainty associated with the forecast. By determining the mean of the probability distribution it is possible to give a deterministic forecast. This highlights the advantages of probabilistic forecasting over deterministic forecasting.

For the probability distribution we are looking for the following two attributes. Firstly, we seek *calibration*, ensuring that actual observations and random draws from the distribution are statistically indistinguishable. Calibration ensures reliable forecasts. Second, we pursue *sharpness*, aiming for forecasts to be as concentrated as feasible. This enhances our certainty regarding specific events. These desired attributes are encapsulated in the paradigm of ”maximizing forecast sharpness while maintaining calibration” [21]. Scoring rules, which will be explained in the following section, are a way of simultaneously assessing sharpness and calibration [22].

3.2 Proper Scoring Rules

This section follows Gneiting and Raftery. [22].

For a deterministic forecast, it is easy to determine the quality of a forecast using an error measure such as the mean squared error. This type of performance measure, which results in a single number, can also be applied to probabilistic forecasts using *scoring rules*. Scoring rules work as follows, denote by $\mathcal{F}$ the set of candidate forecast distributions and let $y \in \mathbb{R}$ be an observation. The scoring rule $S : \mathcal{F} \times \mathbb{R} \to \mathbb{R}$ assigns a numerical value to the forecast $\mathcal{D} \in \mathcal{F}$ given the observation y. In general these scoring rules are negatively oriented, meaning that a lower score indicates better performance. Not every scoring rule is useful, we are interested in a particular set named *proper scoring rules*. Intuitively, these scoring rules encourage a forecaster to provide their honest beliefs and is unbiased. Let Y be the random variable that we are interested in forecasting. A scoring rule S is called proper with respect to $\mathcal{F}$ if

$$\mathbb{E}_G[S(G,Y)] \leq \mathbb{E}_G[S(F,Y)] \tag{3.1}$$

for all $F, G \in \mathcal{F}$, where $\mathbb{E}_G$ denotes the expectation with respect to G. This implies that if the true distribution of Y is G, the best score in expectation is attained in G. The scoring rule is called strictly proper in case there is a strict inequality in equation (3.1) whenever $G \neq F$. Now there is a wide variety of proper scoring rules, one of which is the negative log-likelihood. Another scoring rule is the *Brier score* (BS), which measures the quality of the forecast at a specified threshold $t \in \mathbb{R}$. For a forecast with cumulative distribution function F it is defined by

$$\text{BS}(F, y; t) = (F(t) - \mathbb{1}\{y \leq t\})^2.$$

Here, $\mathbb{1}\{y \leq t\}$ denotes the ideal forecast. Although the Brier score offers valuable insight, it does have a limitation, since it only evaluates at a single threshold. Our interest lies in assessing the overall quality of the forecast. This can be done by integrating the Brier score over all possible thresholds, which leads to the *continuous ranked probability score* (CRPS). This is a proper scoring rule, defined by

$$\begin{aligned} \text{CRPS}(F, y) &= \int_{-\infty}^{\infty} (F(z) - \mathbb{1}\{y \leq z\})^2 \mathrm{d}z \\ &= \mathbb{E}_F(|X - y|) - \frac{1}{2}\mathbb{E}_F(|X - X'|) \end{aligned} \tag{3.2}$$

where X, X' are independent random variables with distribution F [22]. In Figure 3a we can see an example of the computation of the CRPS.

3.3 Weighted Scoring Rules

Scoring rules provide a convenient way to compare two forecasts quantitatively across the full range of possible scenarios. However, in certain scenarios we would like to compare the quality of two forecasts above a certain threshold, so a scoring rule that assigns higher weights to particular events. One can think of different ways of constructing such a scoring rule. The first possibility is to use an outcome-based scoring rule, where additional weighting is performed based on the observation. If we have a weight function $w : \mathbb{R} \to \mathbb{R}$ and a proper scoring rule $S_0 : \mathcal{F} \times \mathbb{R} \to \mathbb{R}$, then we can define a new scoring rule

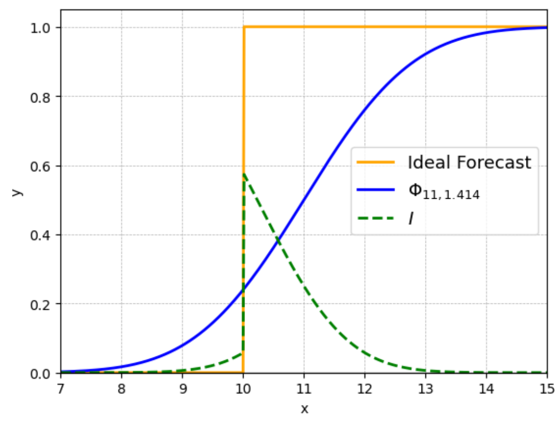
$$S(F, y) = w(y)S_0(F, y). \tag{3.3}$$

However, this scoring rule will be improper, except for when w is constant [13]. This is the case because for the scoring rule S there is a hedging strategy available, since $\mathbb{E}_G[S(F,Y)]$ is not minimal at $F = G$, but at the distribution with density

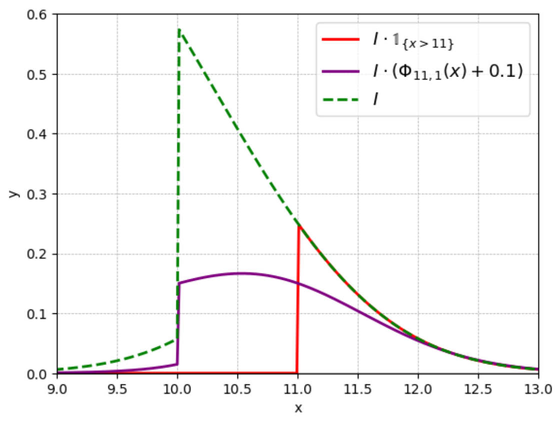
$$f(y) = \frac{g(y)w(y)}{\int g(z)w(z)\mathrm{d}z}.$$

Here $g : \mathbb{R} \to \mathbb{R}$ denotes the density of the true distribution of G. This means that forecasters will deviate from their true beliefs, based on the weighting function w. Therefore it is not recommended to use an outcome-based scoring rule such as in (3.3).

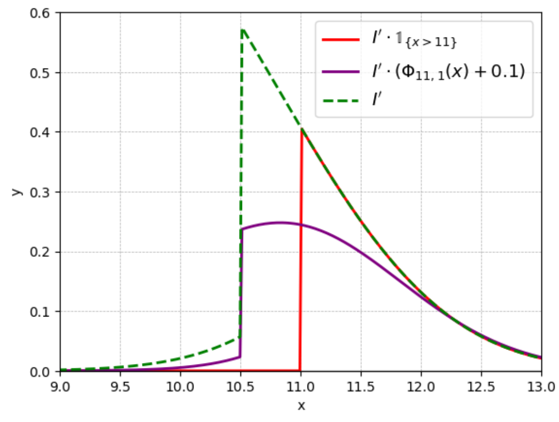
As a result, we need to find alternative methods to incorporate weight functions into scoring rules. The *weighted continuous ranked probability score* (wCRPS) [23] is often used for this. For the weight function



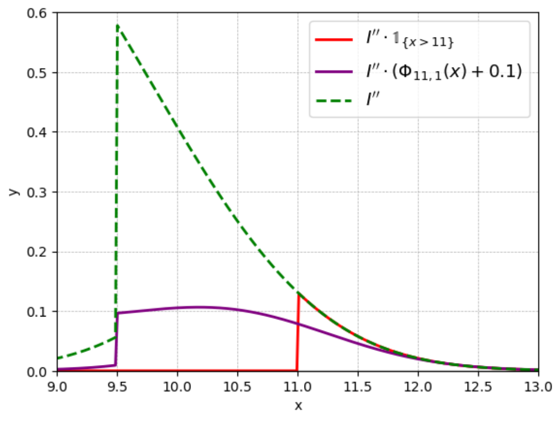
(a) Visualization of the CRPS. Here the forecast distribution is $\Phi_{11,\sqrt{2}}$ and the observation is at 10. The CRPS will be the integral of I.



(b) The effect of the different weight functions, visualizing the indicator weight function with a threshold of 11 and the weight function from (3.6).



(c) The function I' is equal to I shifted 0.5 to the right, meaning that there now is an observation of 10.5 and the forecast distribution is $\Phi_{11.5,\sqrt{2}}$. The weighted scoring rules will assign a higher score to this forecast compared to Figure 3b.



(d) The function I'' is equal to I shifted 0.5 to the left, meaning that there now is an observation of 9.5 and the forecast distribution is $\Phi_{10.5,\sqrt{2}}$. The weighted scoring rules will assign a lower score to this forecast compared to Figure 3b.

Figure 3: The computation of the CRPS and the effect of utilizing different weight functions in the wCRPS. The integrand of (3.2) is denoted by I.

$w : \mathbb{R} \to \mathbb{R}$ the wCRPS is defined by

$$\begin{aligned} \text{wCRPS}(F, z; w) &= \int_{-\infty}^{\infty} (F(z) - \mathbb{1}\{y \le z\})^2 w(z)\mathrm{d}z \\ &= \mathbb{E}_F(|v(X) - v(y)|) - \frac{1}{2}\mathbb{E}_F(|v(X) - v(X')|) \end{aligned} \tag{3.4}$$

where $X, X' \sim F$ are independent and v is any function such that $v(z) - v(z') = \int_{z'}^{z} w(x)\mathrm{d}x$, called the *chaining function* [24]. The proof of the second equality is given in Proposition 1 of [25]. An important property of the wCRPS is that it is a proper scoring rule. The final expression in equation (3.4) provides the option of using a sampling-based approach to estimate the wCRPS. This is useful because for many weight functions there is not an exact expression available.

There are two weight functions that are interesting in our case. First of all, there is the indicator function

$$w(z) = \mathbb{1}\{t \le z\} \tag{3.5}$$

for some threshold $t \in \mathbb{R}$, with chaining function $v(z) = \max(z, t)$. This is an interesting weight function because it provides insight into the performance of the forecast above the threshold t. When this weight function is used the scoring metric will be called the *threshold-weighted continuous ranked probability score* (twCRPS). In the literature this name is also used for the wCRPS, however, we will make a distinction to prevent any confusion. Some examples of where the twCRPS is applied for the evaluation of extreme events are [14, 15, 26, 27]. There are also closed form solutions available for the twCRPS with indicator weight function for multiple distributions [19], which is another advantage.

The second weight function is

$$w(z) = c + \Phi_{\mu,\sigma}(z), \tag{3.6}$$

here $\Phi_{\mu,\sigma} : \mathbb{R} \to \mathbb{R}$ is the cumulative distribution function of a normal random variable with mean μ and standard deviation σ and $c \in \mathbb{R}$. The associated chaining function is [28]

$$v(z) = cz + (z - \mu)\Phi_{\mu,\sigma}(z) + \sigma^2 \phi_{\mu,\sigma}(z), \tag{3.7}$$

where $\phi : \mathbb{R} \to \mathbb{R}$ is the probability density function of a normal random variable with mean μ and standard deviation σ. We verify that this is the chaining function by taking the derivative, finding

$$\begin{aligned}
\frac{d}{dz}\left(cz + (z-\mu)\Phi_{\mu,\sigma}(z) + \sigma^2\phi_{\mu,\sigma}(z)\right) &= c + \left(\Phi_{\mu,\sigma}(z) + (z-\mu)\cdot\frac{d}{dz}\Phi_{\mu,\sigma}(z)\right) + \sigma^2 \cdot \frac{d}{dz}\left(\phi_{\mu,\sigma}(z)\right) \\
&= c + \Phi_{\mu,\sigma}(z) + (z-\mu)\phi_{\mu,\sigma}(z) - (z-\mu)\phi_{\mu,\sigma}(z) \\
&= c + \Phi_{\mu,\sigma}(z)
\end{aligned}$$

where we used the fact that

$$\begin{aligned}
\frac{d}{dz}\left(\phi_{\mu,\sigma}(z)\right) &= \frac{1}{\sqrt{2\pi\sigma^2}} \cdot \left(-\frac{z-\mu}{\sigma^2}\exp\left(-\frac{(z-\mu)^2}{2\sigma^2}\right)\right) \\
&= -\frac{z-\mu}{\sigma^2}\phi_{\mu,\sigma}(z)
\end{aligned}$$

The weight function from equation (3.6) is useful because it gives the user control on how much weight is assigned to lower thresholds compared to higher thresholds in a continuous fashion. This will prove to be useful when we want to produce forecasts that mainly take into account the performance of high thresholds, while not ignoring the performance on lower thresholds, which can be achieved by changing the value of c.

In Figure 3b we can see the effect of using different weight functions. The wCRPS can be obtained by integrating the function from the figure. What is important to note is that in this example, where we use a Gaussian CDF as predictive distribution, the CRPS is shift invariant. This means that if the observation and the mean of the Gaussian CDF are shifted, the CRPS will not change. This is not the case for the wCRPS. In Figures 3c and 3d this is visualized, where both the mean of the forecast distribution and the observation are shifted, so the forecast is of equal performance. If we are shifting to the right, the wCRPS will increase, while if we shift to the left, it will decrease. The CRPS will stay the same in both cases. This shows that the wCRPS is sensitive not only to the position of the observation relative to the predictive distribution but also to where it lies on the x-axis, i.e., whether in an extreme scenario or not. As we shift toward more extreme values, the wCRPS adjusts to reflect their increased significance. This makes the wCRPS an especially valuable tool for evaluating forecast accuracy in situations where correctly predicting extreme events is crucial.

3.4 Verification Measures

The scoring rules from (3.2) and (3.3) provide an overview of the performance of the forecast. However, visual representations can offer a deeper understanding of model quality compared solely using numerical scores. The visualization methods used in this thesis are now discussed, starting with PIT diagrams, followed by reliability diagrams, and as final measure we discuss skill scores.

3.4.1 PIT Diagrams

The first tool is the *probability integral transform* (PIT) diagram. As mentioned in [20], when we have a random variable Y with distribution F, then $F(Y)$ follows a uniform distribution on $[0, 1]$. Hence we can test for calibration by evaluating the CDF of $F(Y)$ using the data. In case the forecast is well calibrated, the resulting CDF will be uniform, meaning that the resulting CDF will be a diagonal line from (0,0) to (1,1).

With PIT diagrams, it is hard to exactly see whether the model is calibrated for extreme events. One tool for this is the *conditional probability integral transform* (cPIT) diagrams [29]. Let Y be a random variable with distribution F, then $Y|Y > t$ is the conditional outcome variable given that the outcome exceeds a specified threshold $t \in \mathbb{R}$, abbreviated by $Y_{>t}$. Computing $F(Y_{>t})$ will not provide any meaningful insight and will not follow a uniform distribution in case $F(t) > 0$. However, by defining the CDF

$$G(x) = \begin{cases} \frac{F(x)-F(t)}{1-F(t)} & \text{if } x > t \\ 0 & \text{if } x \leq t \end{cases}$$

then $G(Y_{>t})$ will follow a standard uniform distribution [29]. Thus, we can compute the conditional PIT values in the following way. First we need to sample from $Y_{>t}$, which is done by taking all observations in the data that exceed t. After this is done, the samples are transformed using distribution function $G(x)$. By then displaying these transformed samples in a diagram, we can gain insight into how well calibrated the forecast is for more extreme events. Ideally, the displayed values should again be uniformly distributed. One important thing to note is that it is possible for a cPIT diagram to seem well calibrated, while over the entire forecast the system is not well calibrated and vice versa. This implies that we cannot only look at a single cPIT diagram and conclude that we have a well calibrated model, but we should also use other verification tools.

3.4.2 Reliability Diagram

With a *reliability diagram* [30], we compare the predicted values with the observations. This is done for a specific threshold $t \in \mathbb{R}$, which means that the problem needs to reduced to a binary classification problem. This will be done similarly to the computation of the Brier score. In this problem class 1 means that the observation exceeds the threshold t and class 0 means that it will not exceed t, so our aim is to predict $\mathbb{1}\{y \geq t\}$. By selecting $p \in [0, 1]$ we can compare the forecasts that assign probability p to observing class 1 compared to the number of observations that are in class 1 for these forecasts. Ideally, the fraction of forecasts that are assigned to class 1 is again equal to p. In a reliability diagram, several values for p are chosen, and these quantities are computed and then displayed in a plot.

In addition to the reliability diagram, we also include the *refinement diagram* [30], which shows the frequencies of forecast probabilities for a specific threshold t. This diagram indicates how often the model predicts certain probabilities for the event occurring at threshold t. This helps provide insight into how the predicted probabilities are distributed, offering a more complete understanding of the model's behavior.

3.4.3 Skill Scores

One tool for comparing the quality of the forecast at a specific threshold t is computing the Brier score at t. It is then possible to compare the performance between different models. This is done with *skill scores* [30], which we will further explain with the Brier score as example. If $\text{BS}_1(t), \text{BS}_2(t) \geq 0$ are the Brier scores for two different models at the same threshold t, then the *Brier skill score* (BSS) at threshold t of model 1 compared to model 2 is defined by

$$\text{BSS}(t) = 1 - \frac{\text{BS}_1(t)}{\text{BS}_2(t)}. \tag{3.8}$$

Here model two serves as a reference model. Note that the scoring rules such as the Brier score are negatively oriented, meaning that a lower score indicates better performance. For this reason, having a positive BSS means that model 1 performs better at the threshold t compared to model 2, while a negative BSS indicates the opposite. Now to gain insight into which model performs better at which thresholds we can compute the BSS for two models for all relevant thresholds t and display this in a single figure, resulting in an easy visual overview into which model performs better at which thresholds.

Similar to the BSS, we can also compute the *threshold-weighted continuous ranked probability skill score* (twCRPSS). This is done using the indicator function with a specific threshold t as weight function, based on equation (3.5). We can then compute the twCRPSS of model 1 compared to model 2 using

$$\mathrm{twCRPSS}(t) = 1 - \frac{\mathrm{twCRPS}_1(t)}{\mathrm{twCRPS}_2(t)}$$

where $\mathrm{twCRPS}_i(t)$ denotes the twCRPS for model $i \in \{1, 2\}$. This score can be interpreted similar to the BSS, except that it now allows to compare the performance of the models above the threshold t instead of at t.

4 Models

This section details the models used for postprocessing NWP forecasts, focusing on the use of EMOS and convolutional neural networks. We will first explain the EMOS algorithm. After that, the choice of parametric distribution will be discussed. Finally, convolutional neural networks are explained, which will be combined with the use of the EMOS algorithm.

4.1 EMOS: Ensemble Setting

We will now provide an overview of the *ensemble member output statistics* (EMOS) model, which was first proposed in [6]. It is one of the most commonly used statistical postprocessing methods, both in academics as well as in operational environments [31]. With EMOS, we assume that a variable of interest, in our case wind speed, follows a specific distribution. The distribution parameters are then estimated based on the ensemble members. This is all done for a fixed lead time. To give a more formal definition, let $\{Z_1, ..., Z_m\}$ denote the individually distinguishable forecasts for a weather variable Y. These forecasts are used to obtain the parameters of a parametric distribution $\mathcal{D}(\mu, \sigma^2)$. Here $\mathcal{D}$ is some specified probability distribution with mean μ and variance σ^2. In the basic EMOS setting, linear regression is used to estimate the parameters of the distributions based on the ensemble members, resulting in

$$\mu = a_0 + \sum_{i=1}^{m} a_i Z_i, \tag{4.1}$$

and

$$\sigma^2 = b_0 + b_1 S_{\text{ens}}^2, \tag{4.2}$$

where

$$S_{\text{ens}}^2 = \frac{1}{n-1} \sum_{i=1}^{m} (Z_i - \bar{Z})^2 \tag{4.3}$$

denotes the ensemble variance and $\bar{Z}$ the ensemble mean. The parameters $a \in \mathbb{R}^{m+1}$ and $b \in \mathbb{R}^2$ are then chosen by fitting the training data using a specific scoring rule. Note that there is a constraint on b, namely $b_i > 0$ for $i \in \{0, 1\}$, ensuring that the variance will be positive.

4.2 EMOS: Deterministic Setting

With the basic framework for EMOS in place, we now turn to the specific adjustments needed to apply EMOS with a determinstic forecast. Instead of using ensemble members $\{Z_1, ..., Z_m\}$ as predictors for Y we will be using a different set of predictors, namely the Harmonie-Arome output for the variables from Table 1, denoted by $\{X_1, ..., X_n\}$. We will keep a linear relation between the predictors and the distribution mean, leading to

$$\mu = a_0 + \sum_{i=1}^{n} a_i X_i \tag{4.4}$$

We explored three different options to estimate the variance, which will be explained in this section. Before we delve into the different methods, the *softplus* activation function is discussed. The softplus function is defined by

$$\text{softplus}(x) = \ln(1 + e^x) \tag{4.5}$$

and it is a smooth approximation of the *rectified linear unit* (ReLU) activation function,

$$\text{ReLU}(x) = \max(0, x). \tag{4.6}$$

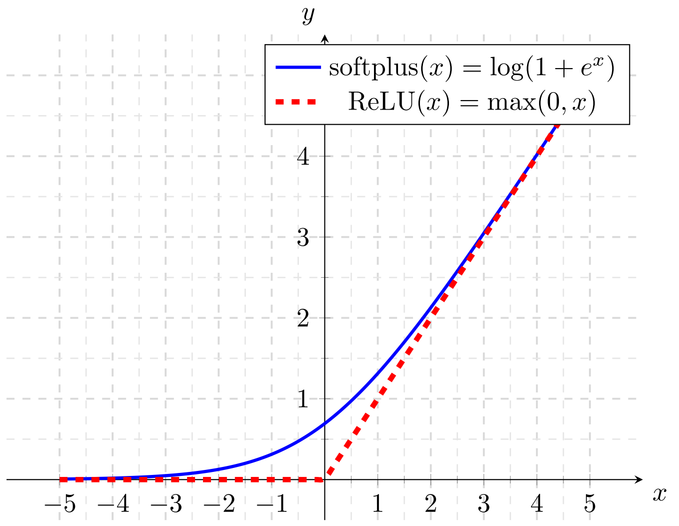


Figure 4: The softplus and ReLU functions from equations (4.5) (4.6).

They can both be used to ensure a variable is non-negative and in Figure 4 a plot of these functions can be seen. We will be using the the softplus function because it is always positive, never reaching zero.

Now the first method of estimating the variance is discussed. We will take the following estimator,

$$\sigma^2 = \text{softplus}\left(c_0 + \sum_{i=1}^{n} c_i X_i\right). \tag{4.7}$$

This equation is similar to (4.4), with a linear relation between the features and the variance. The softplus function is applied to ensure that the variance remains positive. Another option is to use the *spatial variance*. Spatial variance is the variance of the NWP forecast in the grid surrounding the weather station. In case we have chosen a grid of size $2N + 1$ for $N \in \mathbb{N}$, we find

$$S^2_{\text{spatial}} = \sum_{i,j=-N}^{N} (w_{i,j} - \bar{w})^2 \tag{4.8}$$

where the station is located in cell (0,0) and $\bar{w}$ is the mean wind speed in the grid. Using the spatial variance we can then determine σ^2 using an equation similar to (4.2), except that we replace S_{ens} with S_{spatial}. This approach has been used in [8].

The final way to obtain an estimate of the variance is by combining the earlier mentioned approaches. This leads to

$$\sigma^2 = \text{softplus}\left(c_0 + \sum_{i=1}^{n} c_i X_i + c_{n+1} S^2_{\text{spatial}}\right), \tag{4.9}$$

where the softplus function is used again to ensure the variance is positive.

During preliminary testing, it was shown that the first approach, from (4.7), resulted in the best performance. Additional experiments were performed in which we also tested an equation similar to (4.7), except that we used this equation to estimate the standard deviation instead of the variance. Estimating the variance was demonstrated to be better compared to estimating the standard deviation, therefore we proceeded to use formula (4.7) to estimate the variance in all linear models. We will now go more in depth for the different choices of the parametric distribution $\mathcal{D}$.

4.3 Parametric Distribution

One important aspect of EMOS is the choice of parametric distribution. A popular distribution for wind speeds is the truncated normal distribution. In case the main aim is the prediction of extremes, having a more heavy right-tailed distribution could be better because these distributions assign a larger probability to higher wind speeds. Some examples of these distributions are the log-normal distribution and the generalized extreme value distribution. In this section, the choice of parametric distribution will be explored in more detail, including the relation between the distribution parameters and the mean μ and variance σ^2 that are calculated with (4.4) and (4.7).

4.3.1 Truncated Normal Distribution

The *truncated normal* (TN) distribution is the first distribution considered, which is used frequently in the literature for wind speeds [32]. Since we are dealing with wind speeds, we will be truncating the normal distribution from zero to infinity. In two popular deep learning libraries Tensorflow and PyTorch, the truncated normal distribution is implemented by taking the mean and variance of the underlying normal distribution as input parameters. This means that the ”loc” (mean) and ”scale” (standard deviation) provided to the distribution refer to the parameters of the normal distribution before truncation. Consequently, the actual truncated normal distribution will have different mean and variance values that depend on the truncation bounds. In case the truncated bounds are chosen at zero and infinity, this leads to mean and variance

$$\mu + \sigma\frac{\phi(\alpha)}{1-\Phi(\alpha)} \quad \text{and} \quad \sigma^2\left(1+\frac{\alpha\phi(\alpha)}{1-\Phi(\alpha)}-\left(\frac{\phi(\alpha)}{1-\Phi(\alpha)}\right)^2\right)$$

respectively, where μ and σ are the mean and standard deviation of the underlying normal distribution and $\alpha = -\mu/\sigma$.

Given that there is no easy way to account for the correction terms in the mean and variance of the truncated normal distribution, we simply use μ and σ as approximations for the mean and standard deviation.

4.3.2 Log-Normal Distribution

We will now further explore the usage of the *log-normal* (LN) distribution. This approach has already been examined in [15] and [14]. Let $X \sim \mathcal{N}(\mu, \sigma^2)$, then $Y = e^X$ follows a log-normal distribution, denoted by $Y \sim \mathcal{LN}(\mu, \sigma^2)$. We now have

$$\mathbb{E}(Y) = \exp\left(\mu + \frac{\sigma^2}{2}\right) \quad \text{and} \quad \mathbb{V}\text{ar}(Y) = e^{2\mu+\sigma^2}\left(e^{\sigma^2}-1\right).$$

This means that if we want our log-normal distribution to have mean μ_Y and variance σ_Y^2, then we need to have

$$\mu = \log\left(\frac{\mu_Y^2}{\sqrt{\mu_Y^2+\sigma_Y^2}}\right) \quad \text{and} \quad \sigma^2 = \log\left(1+\frac{\sigma_Y^2}{\mu_Y^2}\right).$$

4.3.3 Generalized Extreme Value Distribution

Another notable distribution is the *generalized extreme value* (GEV) distribution. This distribution originates from extreme value theory, where it is used to model the largest (or smallest) value of m samples from some distribution. As m increases, the distribution of the largest of the m samples will converge to the GEV distribution, provided the samples are properly normalized [33]. This result is independent of the distribution from which the m samples are drawn. The GEV has cumulative distribution function

$$F(x) = \begin{cases} \exp\left(-\left(1+\xi_G\left(\frac{x-\mu_G}{\sigma_G}\right)\right)^{-1/\xi_G}\right) & \text{for } \xi_G \neq 0, \text{ and } 1+\xi_G\left(\frac{x-\mu_G}{\sigma_G}\right) > 0 \\ \exp\left(-\exp\left(-\frac{x-\mu_G}{\sigma_G}\right)\right) & \text{for } \xi_G = 0 \end{cases}$$

In this distribution $\mu_G \in \mathbb{R}$ is the location parameter, $\sigma_G > 0$ is the scale parameter, and $\xi_G \in \mathbb{R}$ is the shape parameter.

The parameters μ_G and σ_G^2 will be modelled as the output of a linear model, where the softplus activation function is applied for σ_G^2 to ensure positivity. The final parameter left is ξ_G. This parameter defines the shape of the distribution and determines the tail behavior. Specifically, ξ_G governs the extremeness of the distribution.

- When $\xi_G > 0$, the distribution is *Fréchet-type* (right heavy-tailed).
- When $\xi_G < 0$, the distribution is *Weibull-type* (left heavy-tailed).
- When $\xi_G = 0$, the distribution is *Gumbel-type* (exponential tails).

Again, we use the same approach as in [15], where ξ_G is a parameter that is fitted to the data during training. Once fitted, it remains constant.

4.3.4 Mixture Distributions

It is also possible to combine multiple distributions, to both use the capabilities of the TN for the average predictions and LN or GEV distribution for more extreme forecasts. An example of this is in [27], where a regime switching distribution is employed. In this regime switching distribution, there is a specific threshold $t \in \mathbb{R}$, and if the predicted wind speed from the NWP model is smaller than t, the forecast is produced with a TN, otherwise it is made with a more heavy-tailed distribution, such as the GEV or LN.

Another possibility is with the use of *mixture distributions.* In a simple case, if we have two cumulative distribution functions F_1, F_2 and a weight $w \in [0,1]$ then the mixture distribution F is defined by

$$F(x) = wF_1(x) + (1-w)F_2(x).$$

The weight of the mixture distribution is then a parameter that is fitted using training data. However, similar to the regime switching distribution, we would want one distribution for modeling higher wind speeds and another distribution for modeling lower wind speeds. Since the weight parameter is independent of the input data, this cannot be done with a mixture distribution. One way to make our weight parameter adaptive to the input is the following way, we define our weight parameter by

$$w(X_w) = \text{sigmoid}(\alpha + \beta X_w), \tag{4.10}$$

where the sigmoid function is defined by

$$\text{sigmoid}(x) = \frac{1}{1+e^{-x}}.$$

In equation (4.10) the variable X_w is the NWP forecast of the wind speed and $\alpha, \beta \in \mathbb{R}$ are two parameters that are fitted using training data. Since X_w is a good estimator for the observed wind speed, this is a simple way to determine whether a heavier right-tailed distribution is needed. We refer to this type of distribution as an *adaptive mixture* (AM) distribution.

Regarding the choice of the sigmoid function, it meets the criteria of being a continuous, strictly increasing function with an image in the interval $(0,1)$. Alternatively, other functions, such as the standard Gaussian cumulative distribution function, could also be considered. Another approach is to use additional features to determine the weight parameter. However, we did not explore further experimentation with the Gaussian CDF, alternative functions, or the inclusion of additional features.

4.4 Convolutional Neural Networks

This section follows Goodfellow et al. [34].

Now that we explored EMOS with linear regression, we will look at a more complex model to estimate the parameters of the EMOS distribution. This can namely also be done using *convolutional neural networks* (CNNs). They were introduced by LeCun in 1989 [35] and are a type of neural network specifically designed to handle data with a grid-like structure, such as a grid of pixels for image data. CNNs have achieved significant success in many practical applications. The term ”convolutional neural network” reflects the use of a mathematical operation called convolution.

In the past, CNNs have already succesfully been applied to the postprocessing of weather data, such as in [10] and [9]. In the former it was used on wind speed data in the Netherlands, and in the latter on precipitation in California, showing performance that is similar to other state of the art methods.

In this section, we will provide an overview of CNNs, beginning with an explanation of the convolutional operation. It is assumed that the reader has a basic understanding of deep learning concepts. We will then delve into pooling, a common operation in CNNs, followed by an explanation of batch normalization.

4.4.1 Convolutional Operation

Essentially, convolutional networks are neural networks that incorporate convolution instead of general matrix multiplication in at least one layer. With such a convolutional layer we receive a two-dimensional input $I \in \mathbb{R}^{n\times n}$ and slide a kernel $K \in \mathbb{R}^{m\times m}$ over this input with a convolutional operation, leading to a two-dimensional feature map S defined by

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i-m, j-n)K(m,n) \tag{4.11}$$

Here m and n range over the dimensions of the kernel. The weights of the kernel, $K(m,n)$, are parameters determined by fitting them to a training set. Intuitively, the convolutional operation makes sense, since convolutional networks aim to detect local patterns in the input data by sliding a small window (the kernel) over the input and performing convolution operations. This process allows the network to learn hierarchical representations of features, starting from simple patterns in the lower layers to more complex and abstract features in the deeper layers.

Applying convolution to an input matrix alone reduces the dimensions of the input. To prevent this from happening we can add padding to the input, in which we add zeros to the borders of the input matrix such that the output has the same size as the input. This will also give more importance to the boundaries of the input. In Figure 5 we can see how the convolutional operation works, where an additional padding layer is included to ensure that the output has the same dimensions as the input.

Another aspect of the convolutional layer that we have not addressed yet is the activation function $\sigma : \mathbb{R} \to \mathbb{R}$, such as the ReLu function. This means that the final feature map is defined by

$$S(i,j) = \sigma\left((K * I)(i,j)\right).$$

This introduces non-linearity into the neural network.

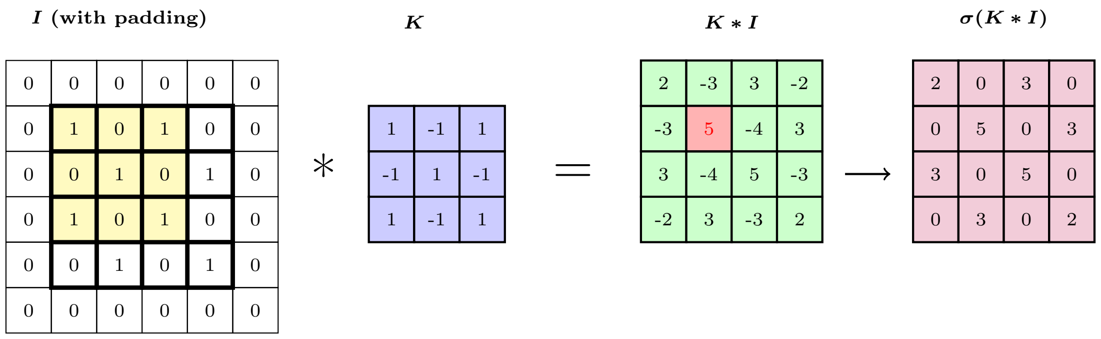


Figure 5: Schematic illustration of a convolutional operation. The highlighted region in the input matrix is convolved with the kernel to produce one element of the output matrix. The activation function is the ReLu.

The convolutional operation helps models learn spatial patterns from the input data, enabling the detection of local features and their hierarchical representation across layers. This capability makes convolutional neural networks effective for tasks like post-processing NWP forecasts, where capturing spatial dependencies is important for accurate predictions.

4.4.2 Pooling

In convolutional neural networks there is often a pooling function after (several) convolutional operations combined with non-linear activation functions. A pooling function is used to reduce the size of the input, while preserving the features. Note that similar to an activation function, this operation does not contain any parameters. A common choice of a pooling function is max pooling, where the maximum value of a square neighbourhood is returned.

4.4.3 Batch Normalization

Batch normalization is a technique to improve the learning of deep neural networks. It is implemented as a separate layer in a neural network and ensures that the layers' output activations follow a unit Gaussian distribution. This helps reduce the effects of problems such as bad weight initialization and the vanishing gradient problem. It is often applied after a convolutional layer.

5 Experiments

The primary objective of these experiments was to determine whether training with the wCRPS as a loss function can enhance the performance of the EMOS model. In addition, we evaluated how various parametric distributions affect model performance. The experiments were conducted on two different versions of EMOS, one utilizing a linear model for the relation between the predictors and the distribution parameters and the other employing convolutional neural networks. With this setup we could test the effect of training models with the wCRPS on a simpler and a more complex model.

All models were trained globally, meaning that a single model was used for all observation stations, instead of fitting a unique model for each station. A sampling-based approach was employed to estimate the loss function values. This meant that we estimated the expected values from equations (3.2) and (3.4) in case of the CRPS and wCRPS respectively. In case a sample size of n was used with a chaining function $v : \mathbb{R} \to \mathbb{R}$, the wCRPS was estimated with

$$\frac{1}{n}\sum_{i=1}^{n}\left(|v(x_i) - v(y)| - \frac{1}{2}|v(x_i) - v(x_i')|\right), \tag{5.1}$$

where x_i and x_i' with $i \in \{1, 2, ..., n\}$ are independently drawn samples from the forecast distribution and y is the observation. In this approach, we drew a new set of samples for each forecast and observation in our training set. To estimate the CRPS with a sampling-based approach, we had a similar expression as in equation (5.1) except we did not transform the samples and observation with the chaining function. To optimize the parameters in our model, we needed to be able to obtain the gradients with respect to random samples drawn from a distribution. This could be done via the *reparametrization trick* [34]. This procedure works by representing a random variable z as a function of a random variable ϵ and model parameters θ, i.e., $z = f(\epsilon, \theta)$, where ϵ follows a fixed distribution. This allows us to propagate the gradients through θ. For example, if $z \sim \mathcal{N}(\mu, \sigma^2)$ we can write $z = \mu + \sigma\epsilon$, where $\epsilon \sim \mathcal{N}(0, 1)$, enabling gradient estimation through μ and σ.

EMOS with linear regression was implemented in TensorFlow [36], with the help of the TensorFlow Probability extension. Additionally, for the CNNs we used Keras [37] as frontend. TensorFlow inherently supports the reparametrization trick within its framework, allowing for efficient gradient computation with random variables. In [38], the full code for this thesis is provided.

This section is organized as follows: first, we describe the linear and CNN models, followed by a discussion of model selection. Finally, we present the results for both linear models and CNNs.

5.1 Linear Regression Description

For EMOS with linear regression, the input consisted of the features from Table 1. For all the features we subtracted the training data mean and divided by the training data standard deviation.

We implemented all parametric distributions from section 4.3, where μ and σ were computed with (4.4) and (4.7). Fixed initialization for the coefficient vectors $a, c \in \mathbb{R}^{n+1}$ was used, setting all their entries to 1. For the GEV distribution there were two possible settings. When ξ_G was initialized to zero, the distribution was and remained of the Gumbel type. In case ξ_G was set to any value other than zero, the distribution switched between the Fréchet and Gumbel type. By experimentation we found that the second case led to better results, thus we initialized ξ_G to 0.3. Every initialization for ξ_G led to convergence, with values between -0.5 and 0.5 leading to faster convergence, thus our choice for 0.3.

For the mixture models, we had three sets of parameters. First, we had μ_1 and σ_1^2, which were determined by (4.4) and (4.7) respectively. They were used to model the mean and variance parameters of the first distribution. Similarly, we had μ_2 and σ_2^2, which were used to model the mean and variance parameters of the second distribution. As a result, we had the associated parameters $a, c, a', c' \in \mathbb{R}^{n+1}$, which were optimized simultaneously. Last of all, we also had a weight parameter. In case of the mixture distribution, this was a parameter in $[0, 1]$, initialized at 0.5, and its constraint was enforced by projecting it back into $[0, 1]$ after every update. For the adaptive mixture distribution, we needed to optimize the parameters $\alpha, \beta \in \mathbb{R}$ from equation (4.10). The idea was that the heavier-tailed distribution, such as the LN or GEV should be used for the higher wind speeds and the TN for lower wind speeds. During training, it was found that this was not always the case. Consequently, we placed additional constraints on α and β. We chose these constraints to be $\alpha \in [4, 12]$ and $\beta \in [-6, -0.6]$, and initialized them at $\alpha = 5$ and $\beta = -1$. Through visual inspection of the weight function we found that these ranges ensured that the first distribution was used for the lower wind speeds and the second distribution for higher wind speeds. The initialization of α and β was set to these values because they satisfied the constraints, and these constraints are enforced by projecting α and β back into their respective ranges after every update.

For the models that had a mixture distribution, we utilized an approach in which we first pre-trained each single parametric distribution for 75 epochs, and these were then used to initialize training of the mixture model. This was done to decrease training times, it did not change the performance of the models. For the linear models, there was no problem with overfitting, which was checked by monitoring the validation loss during cross-validation. Thus, all models were trained until the training loss converged.

We also experimented with random initialization, in which the parameters were initialized to a sample from a standard Gaussian distribution. This did not lead to better results, hence we used fixed initialization since the results were more consistent this way. No alternative distributions for initialization were considered.

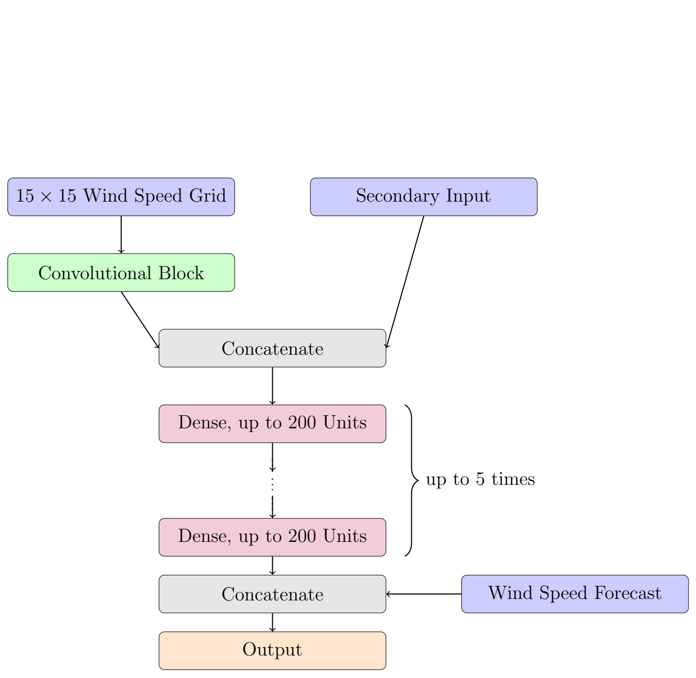
For training, we considered sampling-based approximations of the CRPS and wCRPS as loss functions. For the wCRPS, we experimented with two possible weight functions. The first one was the indicator function from (3.5), and the second weight function was the shifted Gaussian CDF from (3.6). The benefit of the second weight function was that it was highly flexible, since it had multiple parameters, allowing it to balance the weight between extremes and moderate wind speeds. In the sampling-based approach a sample size of 250 was used. The training was also stable for smaller sample sizes, such as 100. The value of 250 was selected to ensure we had consistent and fast convergence of the algorithm.

Next to the hyperparameters in the weight function of the wCRPS and the choice of parametric distribution, there were more hyperparameters that needed to be selected. These hyperparameters were the batch size, the optimizer (Adam or SGD) and the learning rate of the optimizer. These hyperparameters were selected using cross-validation. In section 5.3, we will further elaborate on the validation loss and the algorithm used for hyperparameter optimization.

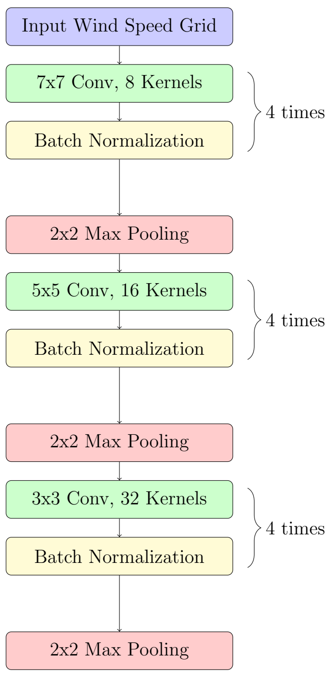
5.2 CNN Description

As mentioned in section 2, the input for the CNN consisted of multiple parts. The first part was a grid from the HA40 wind speed forecasts, with the weather station located in the central grid point. This part of the input was not normalized and was handled by the convolutional part of the neural network. The second part of the input contained the variables from the HA40 forecasts in Table 1, except for the wind speed at 10 meter height. These other four variables were called the *secondary input*. For the secondary input we subtracted the training data mean and divided by the training data standard deviation. This normalization procedure was also applied to the test data, utilizing the mean and standard deviation calculated from the training data. We chose an architecture similar to [8], in which the grid input was handled by a convolutional block and then concatenated with the alternative input. This was then passed to a number of dense layers. The convolutional block had the exact same architecture as in [8]. There was a final part of the input, which was the forecast of HA40 wind speed at the central point of the grid. This input was passed to the final dense layer, before the output of the network was generated. This was added for similar reasons to [8], in which they argue that there is a close to linear relation between the HA40 forecasts and the observations. By passing the HA40 wind speed forecast directly to the final dense layer, the CNN could model the deviation from this linear relation. The complete architecture of the neural network is specified in Figure 6.

The output of the neural network depended on the probability distribution that is chosen. For the TN, the output was the mean and the variance of the normal distribution before it was truncated. The activation function of the mean was linear, and for the variance we used the softplus function. For the LN it contained the mean and variance of the underlying normal distribution, again using the linear and softplus activation function. In case a mixture distribution was used, it contained two sets of parameters of the distributions just mentioned and a weight parameter, with a sigmoid activation function. Since the weight parameter then depended on the input, this can be seen as an extension of the adaptive mixture model. Throughout the rest of the network, the ReLU activation function was used. We also implemented the GEV distribution, where the output of the neural network contained the location, shape and scale parameters of the distribution. However, training the network with the GEV distribution was very unstable, where occasionally the gradients could not be computed, leading the program to crash. Hence we did no further experimentation of the GEV



(a) Overview of the architecture of the CNN.



(b) Convolutional block of the CNN

Figure 6: Architecture of the CNN, where in Figure 6a an overview is given and in 6b the full convolutional block is shown.

distribution in combination with CNNs.

As with the linear models, the models were trained on sampling-based estimates of the CRPS and wCRPS, using equation (5.1). In the sampling-based approach a sample size of 1000 was used. We chose a larger sample size compared to the linear model, since the convergence of CNNs was less stable than for the linear models. For the CNN-based model we chose the largest sample size that still resulted in reasonable training times.

During hyperparameter tuning, we optimized the parameters of the weight function in the wCRPS, the choice of distribution as well as several other parameters: the optimizer (Adam or SGD), the learning rate of the optimizer, the batch size, the number of dense layers, the number of units per dense layer and the level of l_2 penalty for the dense layers.

When CNNs were trained, overfitting was a problem. Accordingly, during validation we utilized early stopping, with a patience of 10 epochs. When training the final models, we used two-thirds of the epochs that yielded the best results during validation, since the full training data set was used to train the final models. This was done by averaging the number of epochs that were deemed optimal on each fold, by training 30 models on each of the three folds.

5.3 Model Selection

We used two different metrics to test the performance of a model during hyperparameter tuning. The first one was the CRPS, which provided an overall performance measure over the entire range of wind speeds. The second scoring metric was the twCRPS with an indicator weight function at threshold 12, denoted by twCRPS12. This scoring metric was particularly valuable because it assessed model performance on extremes. The choice to set the threshold at 12 was based on the distribution of the observations in the training data. It was observed in Table 3 that this threshold approximately corresponded to the 95th percentile of the observations in the training data. Choosing a larger threshold would mean that very few observations exceed it; hence, we settled on a threshold of 12.

During validation, we observed that the CRPS and twCRPS12 were competing objectives. Therefore, the best we could do was to identify a Pareto front containing the best trade-offs. We used the *multiobjective tree-*

structured parzen estimator (MOTPE) [39] for this task. MOTPE is a hyperparameter optimization algorithm that efficiently balances competing objectives by modeling their distributions, enabling the identification of Pareto-optimal solutions that represent the best trade-offs.

The MOTPE implementation from Optuna [40] was used, starting every search with 20 random trials to get a general impression of the search space and model the distribution of the objectives. We then performed around 200 trials with MOTPE for both the linear and CNN models.

5.4 Results Linear Regression

The hyperparameters for the weight function, obtained during hyperparameter tuning can be seen in Table 4 and are illustrated in Figure 7. The indicator weight function is the most extreme, since it solely evaluates the quality of the forecast above 12 m/s. The sharp sigmoid also gives a little bit of weight to lower wind speeds and is smoother, whereas the sigmoid weight has an even more gradual increase. Last of all, we tested the best weight function from CNNs, to see its performance on the linear models. With the exception of the best CNN weight function, all hyperparameters were chosen from the Pareto front identified during the hyperparameter optimization of the linear models. We included the best CNN weight function to test the effect of transferring weight functions between different version of the EMOS model.

Model	**Loss Function**	μ	σ	c	t
Constant Weight	CRPS	-	-	-	-
Indicator Weight	twCRPS	-	-	-	12
Sigmoid Weight	wCRPS	7.05	2.41	0.06	-
Sharp Sigmoid Weight	wCRPS	8.84	1.07	0.02	-
Best CNN Weight	wCRPS	5.42	7.82	0.92	-

Table 4: Selected hyperparameters of the different models. The values μ, σ and c correspond to the parameters from the weighting function from equation (3.6).

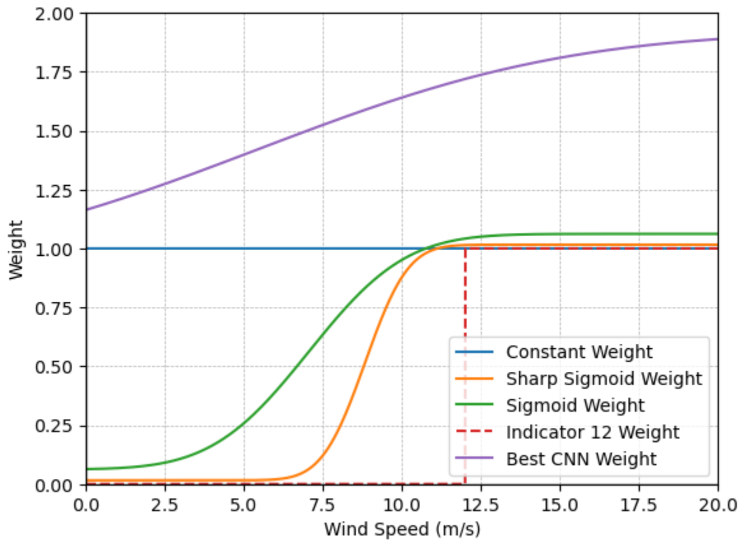


Figure 7: Weight functions for the models from Table 4.

The choice of optimizer and learning rate did not change the results, hence in all models we used Adam with a learning rate of 0.01. The same holds for the batch size, any choice larger than 32 gave the same results, so we fixed it at 256. We chose this value because this resulted in the fastest training times, although the improvement was minor.

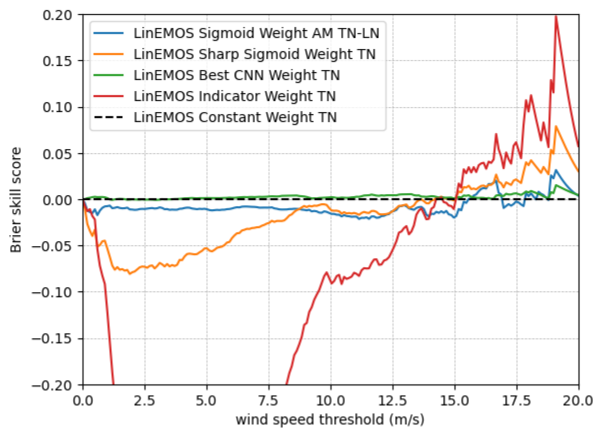
Hyperparameter	**Value**
Optimizer	Adam
Learning Rate	0.02
Batch Size	256

Table 5: Hyperparameters for linear models.

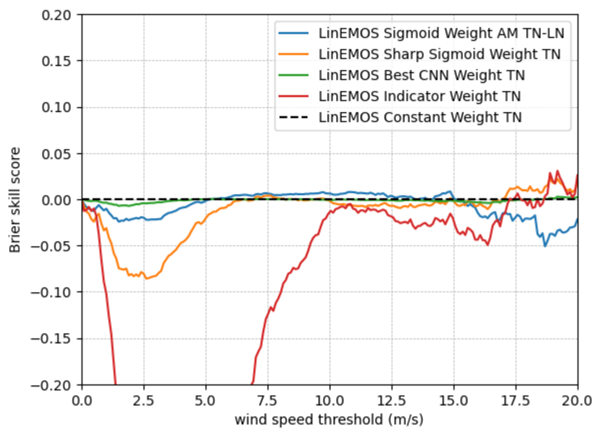
In Figure 8 we can see the Brier skill scores for different model configurations. In this figure the LinEMOS with constant weight and TN as parametric distribution is used as reference model, i.e., a linear EMOS model trained on the CRPS with TN as parametric distribution. Several observations can be made from this figure. First of all, the TN combined with an indicator weight function performs very well above 15 m/s, except CV1. This exception is likely due to the fact that CV1 contains a higher number of extremes in the validation set compared to the training set, as shown in Table 2. The opposite effect happens in CV2, where there are a lot of extremes in the training set compared to the validation set. Although the model trained with the indicator function as weight function performs well above 12 m/s, it exhibits very bad performance below this threshold, where the BSS drops to around -0.2 on CV2 and -0.6 on the test set and CV1. For the two models with a sigmoid function, this worsening in performance for moderate wind speeds is dampened, while this also limits the increase in BSS for higher wind speeds. For the model with the sigmoid weight, the adaptive mixture (AM) distribution was selected by the MOTPE algorithm. The hyperparameters of this model were obtained from the Pareto front with a larger focus on moderate wind speed, compared to the other models. However, this model closely resembles the constant weight with TN in terms of BSS. One final observation is that the best CNN weight function combined with the TN is almost equal in performance to the constant weight TN. In Figure 9 we can see the standard deviation of the BSS by bootstrapping the test data 10.000 times, which showcases that there is a lot of uncertainty above 15 m/s. The result of improved performance on extremes at the cost of worse performance on moderate wind speeds is in line with the results from [19], in which the authors call this phenomenon a body-tail trade-off.

Figure 10 shows a variety of cPIT, sharpness and reliability diagrams. Based on the PIT diagram from 10a and reliability and sharpness diagram with a threshold of five in Figure 10c, we can see that especially the model with the indicator weight is not well calibrated. It has a positive bias, and the sharpness diagram shows that it consistently predicts too high wind speeds. A similar, although less pronounced, pattern can be observed for the model using the sharp sigmoid weight function. The other models have very similar calibration at a threshold of 5 m/s, with only a slight positive bias. The calibration changes when we look at higher thresholds, since in the cPIT, reliability and sharpness diagrams with a threshold of 12 m/s in Figures 10b and 10d, it can be observed that all models are all well calibrated, and there is no significant difference between the models.

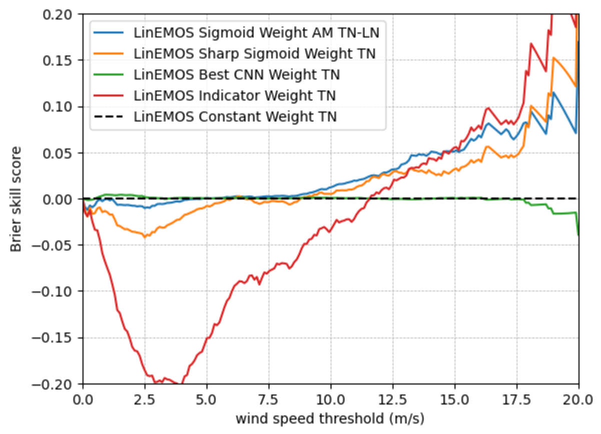
Last, we look at the effect of choosing a different parametric distribution. Figure 11 displays the Brier skill scores of the TN, LN and GEV trained on the constant and indicator weight functions. With a constant weight function, the LN and GEV distributions are slightly better between 1 and 12 m/s, with worse performance for extremes. The GEV model performs very poorly below 1 m/s, because it can predict negative wind speeds, something that does not occur for the LN and TN. When we train on the indicator function, the GEV distribution performs extremely bad for lower wind speeds, and its BSS curve is not visible until approximately 8 m/s. Above 15 m/s, the TN, LN and GEV combined with the indicator weight all have very similar BSS.



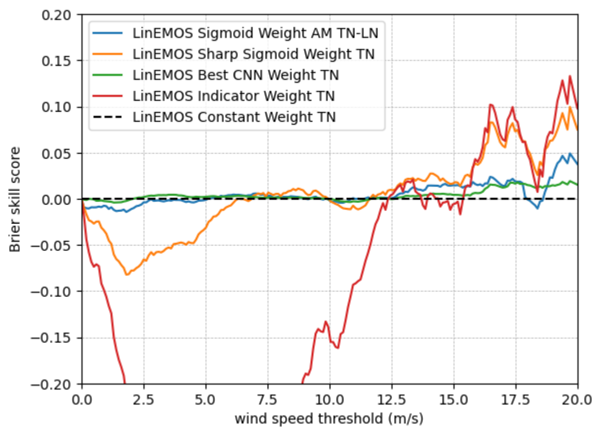
(a) Brier skill scores on the test set.



(b) Brier skill scores on CV1.



(c) Brier skill scores on CV2. The indicator weight with TN reaches a BSS of 0.25 at 19 m/s.



(d) Brier skill scores on CV3.

Figure 8: Brier skill scores for different model configurations on the test set and for cross-validation. CVi denotes that fold i is left out for validation, while the other two folds are used to train. The BSS of the indicator weight with TN drops to around −0.6.

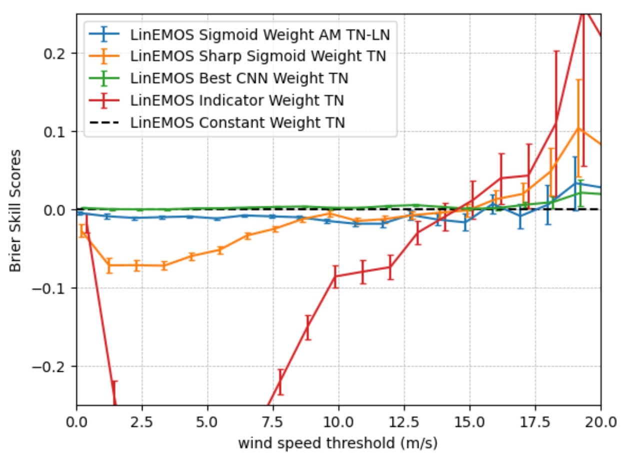
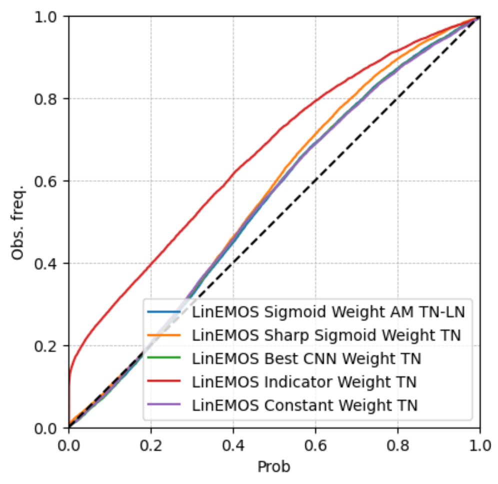
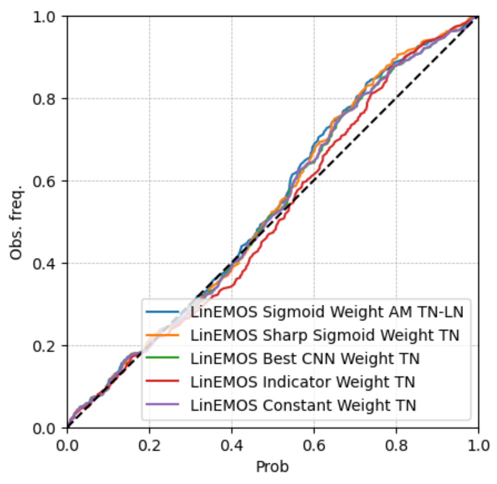


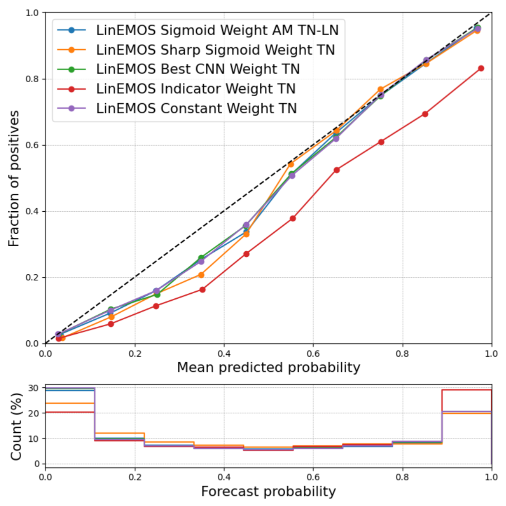
Figure 9: Bootstrapped Brier skill scores on the test set, with a bootstrap size of 10.000.



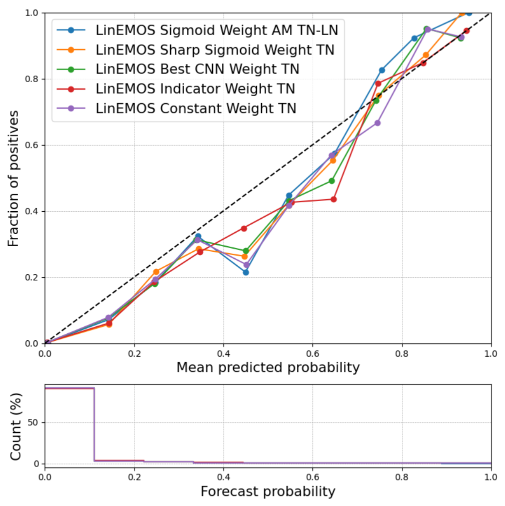
(a) PIT diagram.



(b) cPIT diagrams with a threshold of 12.



(c) Reliability and sharpness diagram at a threshold of 5.



(d) Reliability and sharpness diagram at a threshold of 12.

Figure 10: Calibration of the different models on the test set.

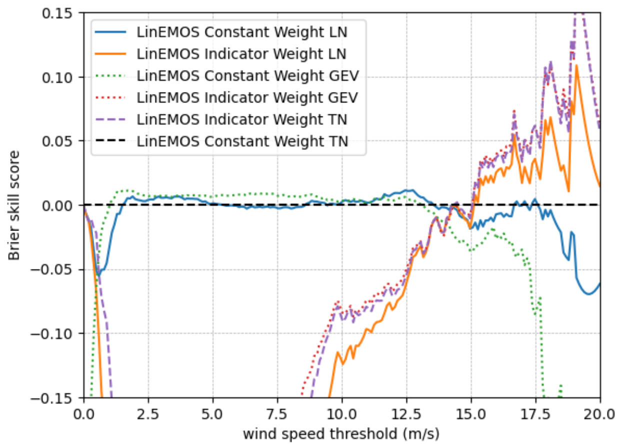
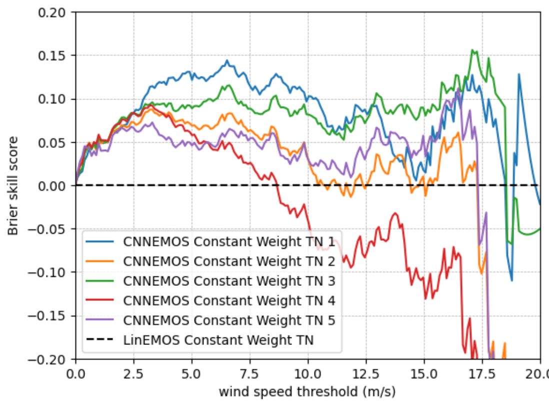


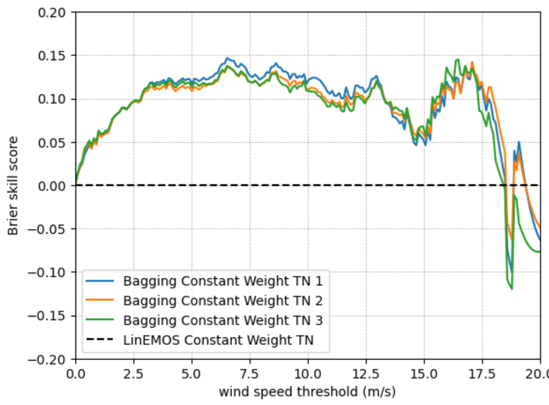
Figure 11: Brier skill scores on the test set with different distributions.

5.5 Results CNNs

For CNNs, a significant amount of variability in performance was observed between different training runs. There were three different sources of this randomness; the first one was the batch selection in optimization, the second one was the sampling-based approach in estimating the CRPS and wCRPS, and the last source was the random parameter initialization. This made comparing single CNNs misleading. This variability of different runs is showcased in Figure 12a. As a solution, we used a bagging version of CNNs, where we trained 10 models with the exact same hyperparameters, and then combined the 10 different predictive distributions into a single distribution by taking their average. The resulting distribution was then used to compute the different scores, such as the Brier score. Three different runs of the bagging model are shown in Figure 12b. In Figures 18a and 18b in Appendix A the PDFs of the bagging estimator are shown for two different samples from the test data. In certain scenarios, the individual models all agreed on what the distribution parameters were, while in other cases they had varying predictions.



(a) Brier skill score of single CNNs, all trained with the same hyperparameters.



(b) Brier skill score of bagging 10 CNNs, all trained with the same hyperparameters.

Figure 12: Brier skill scores of the bagging and single CNN models.

For hyperparameter optimization, we employed a two-step approach. In the first step, we used a broader search space, to optimize the parameters listed in Table 6a, along with the weight function and the parametric distribution. The search space is shown in Table 7 in Appendix A. During this phase, the hyperparameters in Table 6a were much more influential than the choice of weight function and distribution. Therefore, after this first round of optimization, we fixed all the hyperparameters mentioned in Table 6a and then performed another round of hyperparameter optimization, focusing solely on the weight function and distribution. This was done to ensure that we had a fair comparison between all the different models.

For the final models, the number of epochs during training was varied per model, since certain combinations of weight functions and distribution required more epochs to train during validation. These values can be seen in Table 6b. For the models in Table 6b, the Bagging Constant Weight TN model served as a reference model. The parameters from the Best CNN Weight function were selected by the MOTPE algorithm as one of the Pareto optimal solutions. Both the TN and the mixture distribution of TN-LN were frequently selected as a good distribution, which is why we included both distributions in the final model. Since the LN distribution was not selected in the best configurations of the MOTPE algorithm, we did not train any final model with this distribution. The model with the Sharp Sigmoid Weight function was included to demonstrate the impact of applying a weight function that performed well for linear models to CNNs.

The Brier skill scores can be seen in Figures 13 and 14. In both figures, a bagging model trained on the CRPS with TN as parametric distribution is used as reference model. For the best CNN weight function, there is no noticeable body-tail trade-off, since the scores are approximately equal until 12 m/s. Above this threshold, the best CNN weight function combined with the TN and the mixture distribution of TN and LN both perform very similarly, with an improvement over the constant weight function. The sharp sigmoid function, a good weight function for linear models, performs very poorly. Only at about 14 m/s is the BSS close to the constant weight function, while being significantly worse for all other thresholds.

Hyperparameter	Value
Optimizer	Adam
Learning Rate	0.000105
Batch Size	64
Dense Layers	2
Units per Dense Layer	170
l_2 Penalty	0.031658

(a) Hyperparameters for CNNs.

Model Name	Epochs
Bagging Constant Weight TN	47
Bagging Best CNN Weight TN	47
Bagging Best CNN Weight TN-LN	50
Bagging Sharp Sigmoid Weight TN	49

(b) Number of epochs used to train the different models.

Table 6: Hyperparameters and number of epochs used in the CNNs.

Figure 15 displays the twCRPSS scores for the bagging models, with a model trained on the CRPS with TN distribution as reference. The twCRPS at a threshold of 12 m/s for the models with the best CNN weight function is clearly better than the model with a constant weight function. Above 17 m/s, the twCRPS becomes erratic. This can be due to the sampling-based approach, which makes estimating the tails of the distribution more challenging. Additionally, the limited data in this region further complicates the estimation.

Finally, figure 16 contains the BSS for both the linear as well as the bagging models with climatology as reference model. Climatology is the probabilistic forecast based solely on past measurements, and a separate climatology model is constructed for each observation station. Between 0 and 12 m/s the bagging models with constant weight and best CNN weight are the best performers. Between 12 and 17 m/s the bagging models with the best CNN weight have the best results, and above 17 m/s the linear EMOS model with indicator weight has the best BSS.

In terms of calibration, all models, except the model that utilizes a sharp sigmoid weight function, perform equally well. This can be seen in Figure 17, which contains cPIT, sharpness and reliability diagrams. In line with the linear models, there is not a big difference in performance between the model using a TN and the models using a mixture distribution of TN-LN.

For the linear models with the adaptive mixture distribution a constraint was used since the models were not consistently choosing the TN for the lower wind speeds and the GEV or LN for higher wind speeds. For the mixture distributions with CNNs we had a similar result. In figures 20a and 20b from appendix A we can see how much weight is assigned to the TN compared to the LN on the test set. We see strongly varying results, where in certain models only the TN or LN is used for extremes, and in other models the weight is close to a half for all the samples in the test set.

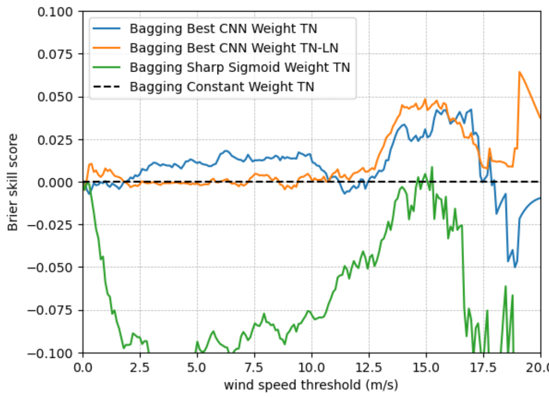


Figure 13: Brier skill score on the test data.

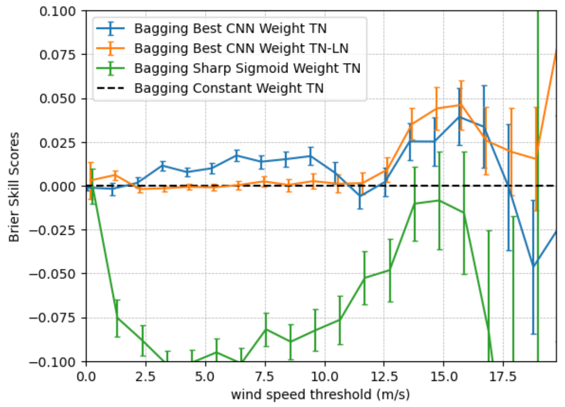


Figure 14: Bootstrapped Brier skill scores on the test data, with a bootstrap size of 10.000.

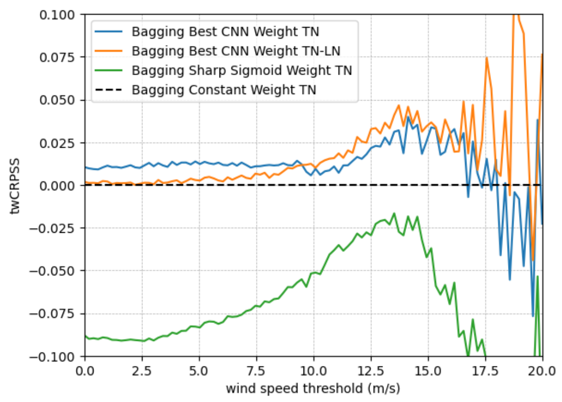


Figure 15: twCRPSS on the test data.

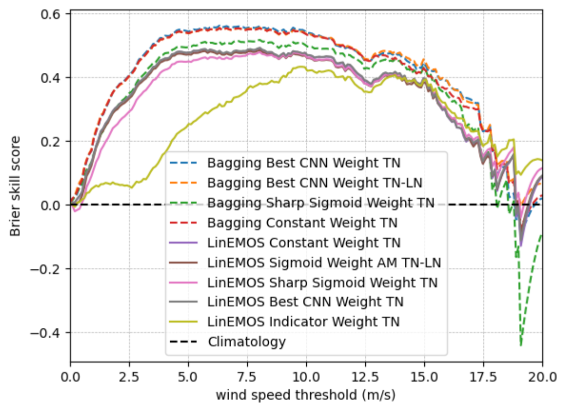
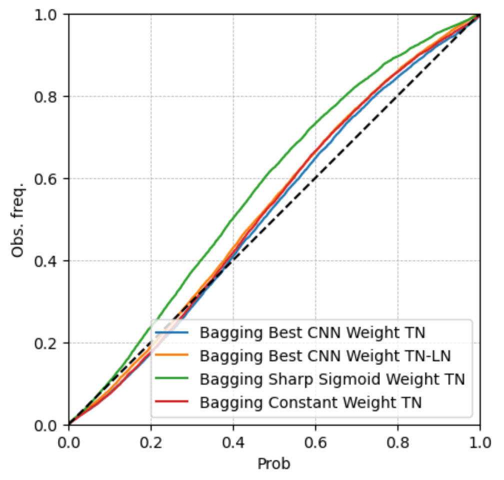
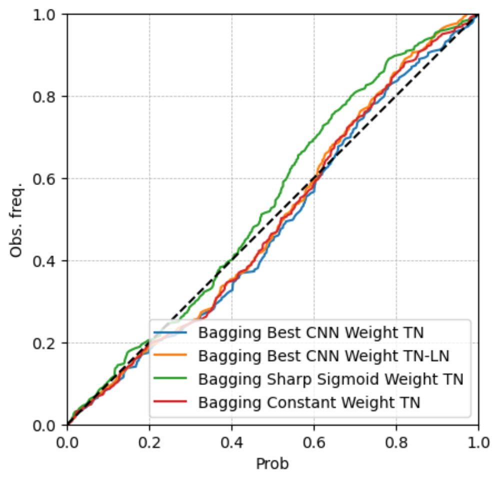


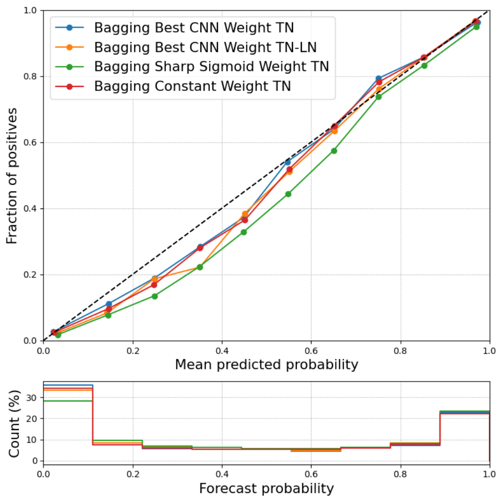
Figure 16: Brier skill scores on the test data with climatology as reference.



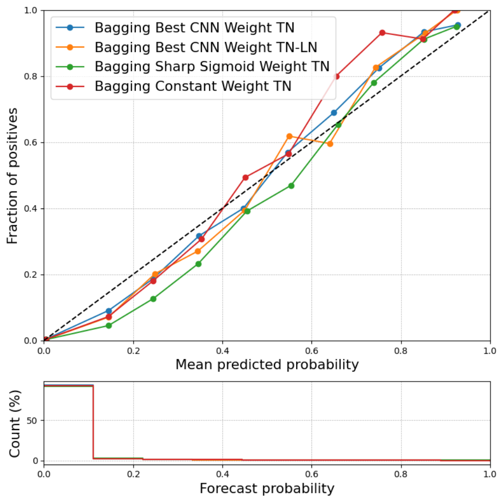
(a) PIT diagram.



(b) cPIT diagrams with a threshold of 12.



(c) Reliability and sharpness diagram at a threshold of 5.



(d) Reliability and sharpness diagram at a threshold of 12

Figure 17: Calibration of the bagging models on the test set.

6 Conclusion and Discussion

In this thesis, we examined the impact of training two versions of the EMOS model using the wCRPS as the loss function instead of the traditional CRPS: one version utilizing linear regression and the other incorporating CNNs. It was demonstrated that using the wCRPS as a loss function could improve performance on high wind speeds in both linear models and CNNs, provided that the weight function was correctly chosen. For linear models, we observed a body-tail trade-off, where improved performance on extremes resulted in reduced performance on the central body of the distribution. To mitigate this problem, we suggested using a class of shifted Gaussian CDFs, to tune how much weight was assigned to extremes compared to averages. This, in combination with a multi-objective hyperparameter optimization algorithm such as MOTPE, was shown to be effective in giving the user control on the performance of extremes relative to the performance on the averages. For CNNs, this body-tail trade-off was not prevalent. Using the optimal weight function for CNNs only led to increased performance on extremes. It is important to note that we performed all the testing of CNNs on bagging versions, since these were able to provide consistent results and were less noisier than the single CNNs.

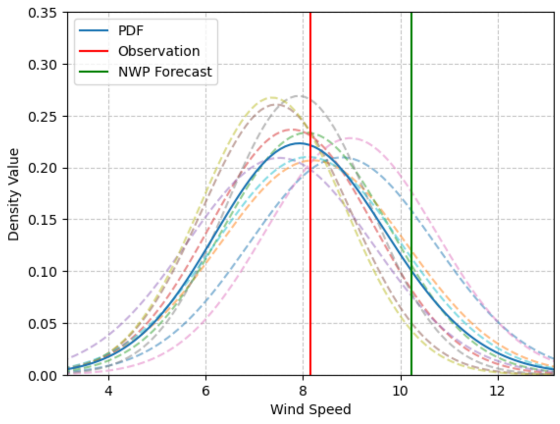
Another observation was that the optimal weight function does not transfer between models. When the best weight function from linear models was used to train CNNs, performance deteriorated for both averages and extremes. Conversely, using the best CNN weight function on linear models, resulted in a model that was almost identical to a model trained on a constant weight function, i.e., the CRPS. This indicated that the best choice of weight function in the wCRPS is model-specific.

Additionally, the best choice of parametric distribution was considered for both the linear models as well as the CNNs. For this data set, the findings indicated that choosing a different distribution did not lead to better performance, and the models did not consistently choose a specific distribution for extremes. This was the case for both the linear models and the CNNs. For both models choosing a different distribution did not affect the performance and in case a mixture distribution was used, the decision on which distribution to use for higher and lower wind speeds seemed random. There can be two reasons why we did not observe any significant difference in the choice of distribution. The first reason is the fact that we used a sampling-based approach. This made the gradients noisier, and estimating the tail behavior of the distribution could have been especially hard since sampling from the tails is unlikely. The second cause is that our data set contained ten-minute average wind speeds, which could benefit the TN. It also can be the case that the choice of distribution is not very important in EMOS models, compared to the choice of loss function and the underlying model.

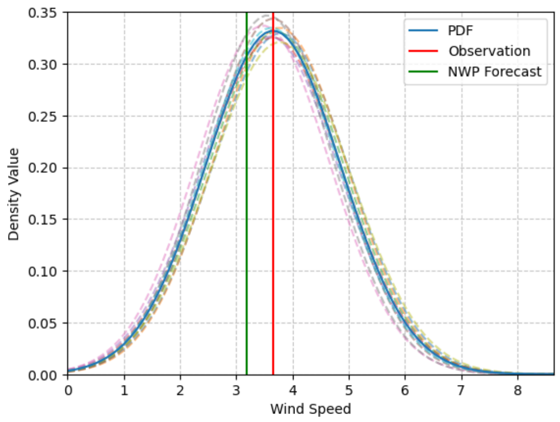
Although using wCRPS can improve performance with the correct choice of weight function, it should first be evaluated whether this is the best approach to take. Looking for the correct weight function with an appropriate optimization algorithm is challenging, since a lot of models have to be trained to find a good set of hyperparameters. For instance, the Brier skill score for a CNN-based model compared to a linear model reference ranged from 0.1 to 0.15, indicating a substantial difference, as shown in Figure 12b. The increase in Brier skill score for the linear models that solely focus on extremes varied between 0.025 and 0.1, while these models had extremely poor performance on the average wind speeds. For CNNs, the Brier skill score of a model with a correctly chosen weight compared to a constant weight function was also approximately 0.05. Therefore it should be carefully considered where the resources are used, since the change from a simple to a more sophisticated model was much greater than training on the wCRPS.

There are several subjects that can be further investigated in future work. First of all, the choice of weight function for CNNs should be studied with a larger search space. In our search space, which contained shifted Gaussian CDFs the optimal weight function resembled an affine function. Therefore, looking at the class of affine functions, or even quadratic function can be interesting. It can be worthwhile investigating these classes of functions, given the optimal weight function that we found. Another promising area of investigation is exploring different weighted scoring rules. We only considered the wCRPS, but there are different weighted scoring rules to consider. Examples are the outcome-weighted CRPS [41], and the vertically re-scaled CRPS [25]. For the outcome-weighted CRPS the CRPS is weighted by the outcome and to ensure a proper scoring rule the forecast is evaluated by its weighted representation. The vertically re-scaled CRPS weights the distance between the forecast and the observation, again a different approach from the wCRPS. In [29] these other weighted variations of the CRPS are discussed, as well as their benefits and drawbacks. It would be interesting to see if the optimal choice of weight function differs between weighted scoring rules. Another topic is to see whether the optimal weight function for the wCRPS can be determined analytically for a linear model. This can help with the understanding of the effect of using a weighted scoring rule as loss function. A final method that can be considered is utilizing importance sampling, where the extremes are being upsampled. This can cause the model to see the extreme observations more often during training, which can help improve predictions on extremes. In this project we did some testing using importance sampling, and it was shown to be effective in decreasing training times for EMOS with linear regression. For EMOS with CNNs the results were much more complicated, where the results differed on the weight function that was selected.

A Appendix

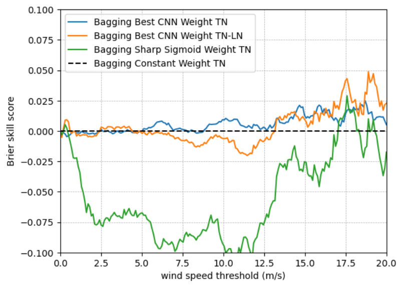


(a) The individual models have varying predictions.

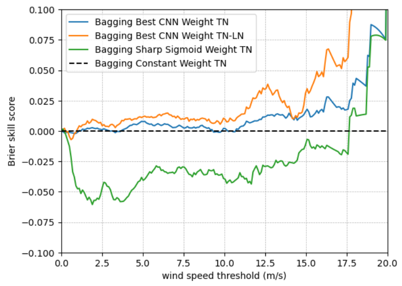


(b) The individual models all have close to the same prediction.

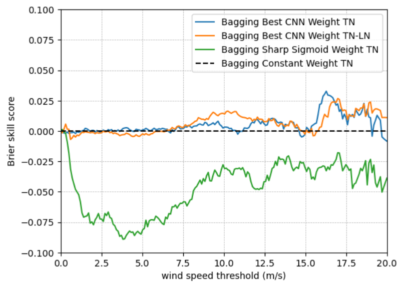
Figure 18: Two forecasts of the CNN combined with bagging. The dashed lines show the predictions of individual CNNs.



(a) Brier skill score of the bagging models on CV1.



(b) Brier skill score of the bagging models on CV2.

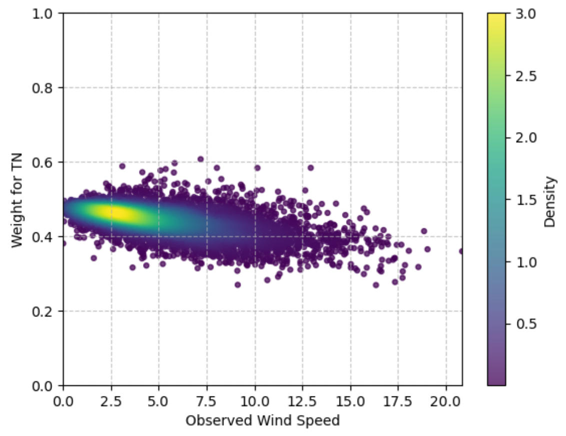


(c) Brier skill score of the bagging models on CV3.

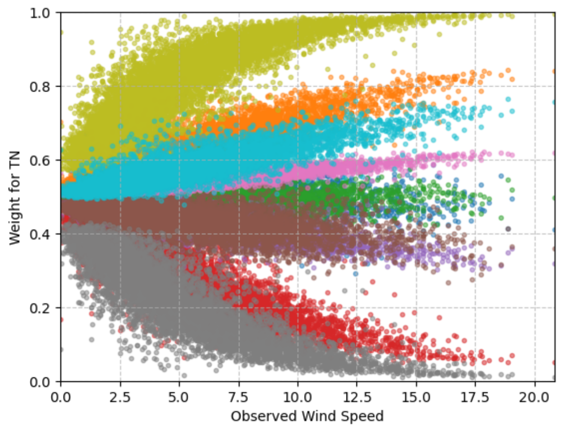
Figure 19: Brier skill scores of the bagging models for cross-validation.

Hyperparameter	**Range**
Parametric Distribution	{Truncated Normal, Log-Normal, Mixture}
$\boldsymbol{\mu}$ (Weight Function)	[-5, 15]
$\boldsymbol{\sigma}$ (Weight Function)	[0.0001, 10] (log scale)
$\boldsymbol{c}$ (Weight Function)	[0.000001, 1]
Optimizer	{Adam, SGD}
Learning Rate	[0.0001, 0.03]
l_2 Penalty	[0.00005, 0.1] (log scale)
Dense Layers	[1, 5]
Units per Dense Layer	[30, 200] (step size 10)
Batch Size	{16, 32, 64, 128, 256, 512, 1024}

Table 7: Search space for CNN hyperparameters.



(a) Weight for the TN distribution in a single EMOS model with CNNs.



(b) Weight for the TN distribution in a TN-LN mixture distribution for all 10 EMOS models from a bagging model.

Figure 20: Weight for the TN distribution in a TN-LN mixture distribution.

B Appendix

Location	**Code**	**Lon (E°)**	**Lat (N°)**
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 (TERSCHLING)	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	6.197	53.412
HEINO	278	6.259	52.434
HOOGEVEEN	279	6.540	52.731
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	308	6.891	52.273
CADZAND	308	3.379	51.380
VLISSINGEN	310	3.603	51.451
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	3.841	51.225
WILHELMINADORP	323	3.894	51.530
STAVENISSE	324	4.006	51.568
HOEK VAN HOLLAND	330	4.122	51.991
THOLEN	331	4.192	51.479
WOENSDRECHT	340	4.342	51.448
R'DAM-GEULHAVEN	343	4.433	51.892
ROTTERDAM	344	4.435	51.925
CABAUW	348	4.936	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	5.788	51.174
MAASTRICHT	380	5.762	50.905
ARCEN	391	6.196	51.497

Table 8: Locations of the stations with corresponding codes and coordinates.

References

[1] Eugenia Kalnay. *Atmospheric Modeling, Data Assimilation and Predictability.* Cambridge University Press, 2002.

[2] Lisa Bengtsson et al. “The HARMONIE–AROME Model Configuration in the ALADIN–HIRLAM NWP System”. In: *Monthly Weather Review* 145.5 (2017), pp. 1919–1935. DOI: 10.1175/MWR-D-16-0417.1. URL: https://journals.ametsoc.org/view/journals/mwre/145/5/mwr-d-16-0417.1.xml.

[3] Stéphane Vannitsem et al. “Statistical Postprocessing for Weather Forecasts: Review, Challenges, and Avenues in a Big Data World”. In: *Bulletin of the American Meteorological Society* 102.3 (2021), E681–E699. DOI: 10.1175/BAMS-D-19-0308.1. URL: https://journals.ametsoc.org/view/journals/bams/102/3/BAMS-D-19-0308.1.xml.

[4] M. Leutbecher and T.N. Palmer. “Ensemble forecasting”. In: *Journal of Computational Physics* 227.7 (2008). Predicting weather, climate and extreme events, pp. 3515–3539. ISSN: 0021-9991. DOI: https://doi.org/10.1016/j.jcp.2007.02.014. URL: https://www.sciencedirect.com/science/article/pii/S0021999107000812.

[5] Tilmann Gneiting and Adrian E. Raftery. “Weather Forecasting with Ensemble Methods”. In: *Science* 310.5746 (2005), pp. 248–249. DOI: 10.1126/science.1115255. eprint: https://www.science.org/doi/pdf/10.1126/science.1115255. URL: https://www.science.org/doi/abs/10.1126/science.1115255.

[6] Tilmann Gneiting et al. “Calibrated Probabilistic Forecasting Using Ensemble Model Output Statistics and Minimum CRPS Estimation”. In: *Monthly Weather Review* 133.5 (2005), pp. 1098–1118. DOI: 10.1175/MWR2904.1. URL: https://journals.ametsoc.org/view/journals/mwre/133/5/mwr2904.1.xml.

[7] Stephan Rasp and Sebastian Lerch. “Neural Networks for Postprocessing Ensemble Weather Forecasts”. In: *Monthly Weather Review* 146.11 (2018), pp. 3885–3900. DOI: 10.1175/MWR-D-18-0187.1. URL: https://journals.ametsoc.org/view/journals/mwre/146/11/mwr-d-18-0187.1.xml.

[8] Daniel Teixeira Soares Tolomei. “Multivariate Postprocessing of Temporal Dependencies with Autoregressive and LSTM Neural Networks”. In: (2022). URL: https://studenttheses.uu.nl/handle/20.500.12932/41500.

[9] Michael Scheuerer et al. “Using Artificial Neural Networks for Generating Probabilistic Subseasonal Precipitation Forecasts over California”. In: *Monthly Weather Review* 148.8 (2020), pp. 3489–3506. DOI: 10.1175/MWR-D-20-0096.1. URL: https://journals.ametsoc.org/view/journals/mwre/148/8/mwrD200096.xml.

[10] Simon Veldkamp et al. “Statistical Postprocessing of Wind Speed Forecasts Using Convolutional Neural Networks”. In: *Monthly Weather Review* 149.4 (2021), pp. 1141–1152. DOI: 10.1175/MWR-D-20-0219.1. URL: https://journals.ametsoc.org/view/journals/mwre/149/4/MWR-D-20-0219.1.xml.

[11] Kevin Höhlein et al. “Postprocessing of Ensemble Weather Forecasts Using Permutation-Invariant Neural Networks”. In: *Artificial Intelligence for the Earth Systems* 3.1 (2024), e230070. DOI: 10.1175/AIES-D-23-0070.1. URL: https://journals.ametsoc.org/view/journals/aies/3/1/AIES-D-23-0070.1.xml.

[12] Nina Horat and Sebastian Lerch. “Deep Learning for Postprocessing Global Probabilistic Forecasts on Subseasonal Time Scales”. In: *Monthly Weather Review* 152.3 (2024), pp. 667–687. DOI: 10.1175/MWR-D-23-0150.1. URL: https://journals.ametsoc.org/view/journals/mwre/152/3/MWR-D-23-0150.1.xml.

[13] Sebastian Lerch et al. “Forecaster’s Dilemma: Extreme Events and Forecast Evaluation”. In: *Statistical Science* 32.1 (2017), pp. 106–127. DOI: 10.1214/16-STS588. URL: https://doi.org/10.1214/16-STS588.

[14] Sándor Baran and Sebastian Lerch. “Log-normal distribution based Ensemble Model Output Statistics models for probabilistic wind-speed forecasting”. In: *Quarterly Journal of the Royal Meteorological Society* 141.691 (Mar. 2015), pp. 2289–2299. ISSN: 1477-870X. DOI: 10.1002/qj.2521. URL: http://dx.doi.org/10.1002/qj.2521.

[15] Sebastian Lerch and Thordis L. Thorarinsdottir. "Comparison of non-homogeneous regression models for probabilistic wind speed forecasting". In: *Tellus A: Dynamic Meteorology and Oceanography* 65.1 (Dec. 2013), p. 21206. ISSN: 1600-0870. DOI: 10.3402/tellusa.v65i0.21206. URL: http://dx.doi.org/10.3402/tellusa.v65i0.21206.

[16] Sándor Baran, Patrícia Szokol, and Marianna Szabó. "Truncated generalized extreme value distribution-based ensemble model output statistics model for calibration of wind speed ensemble forecasts". In: *Environmetrics* 32.6 (Apr. 2021). ISSN: 1099-095X. DOI: 10.1002/env.2678. URL: http://dx.doi.org/10.1002/env.2678.

[17] Philipp Hess and Niklas Boers. "Deep Learning for Improving Numerical Weather Prediction of Heavy Rainfall". In: *Journal of Advances in Modeling Earth Systems* 14.3 (2022). e2021MS002765 2021MS002765, e2021MS002765. DOI: https://doi.org/10.1029/2021MS002765. eprint: https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1029/2021MS002765. URL: https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2021MS002765.

[18] D. R. Scheepens et al. "Adapting a deep convolutional RNN model with imbalanced regression loss for improved spatio-temporal forecasting of extreme wind speed events in the short to medium range". In: *Geoscientific Model Development* 16.1 (2023), pp. 251–270. DOI: 10.5194/gmd-16-251-2023. URL: https://gmd.copernicus.org/articles/16/251/2023/.

[19] Jakob Benjamin Wessel et al. *Improving probabilistic forecasts of extreme wind speeds by training statistical post-processing models with weighted scoring rules.* 2024. arXiv: 2407.15900 [cs.LG]. URL: https://arxiv.org/abs/2407.15900.

[20] Tilmann Gneiting and Matthias Katzfuss. "Probabilistic Forecasting". In: *Annual Review of Statistics and Its Application* 1.Volume 1, 2014 (2014), pp. 125–151. ISSN: 2326-831X. DOI: https://doi.org/10.1146/annurev-statistics-062713-085831. URL: https://www.annualreviews.org/content/journals/10.1146/annurev-statistics-062713-085831.

[21] Tilmann Gneiting, Fadoua Balabdaoui, and Adrian E. Raftery. "Probabilistic Forecasts, Calibration and Sharpness". In: *Journal of the Royal Statistical Society Series B: Statistical Methodology* 69.2 (Mar. 2007), pp. 243–268. ISSN: 1369-7412. DOI: 10.1111/j.1467-9868.2007.00587.x. eprint: https://academic.oup.com/jrsssb/article-pdf/69/2/243/49794500/jrsssb_69_2_243.pdf. URL: https://doi.org/10.1111/j.1467-9868.2007.00587.x.

[22] Tilmann Gneiting and Adrian E Raftery. "Strictly Proper Scoring Rules, Prediction, and Estimation". In: *Journal of the American Statistical Association* 102.477 (2007), pp. 359–378. DOI: 10.1198/016214506000001437. eprint: https://doi.org/10.1198/016214506000001437. URL: https://doi.org/10.1198/016214506000001437.

[23] Tilmann Gneiting and Roopesh Ranjan. "Comparing Density Forecasts Using Threshold- and Quantile-Weighted Scoring Rules". In: *Journal of Business & Economic Statistics* 29.3 (2011), pp. 411–422. DOI: 10.1198/jbes.2010.08110. eprint: https://doi.org/10.1198/jbes.2010.08110. URL: https://doi.org/10.1198/jbes.2010.08110.

[24] Maxime Taillardat et al. "Evaluating probabilistic forecasts of extremes using continuous ranked probability score distributions". In: *International Journal of Forecasting* 39.3 (2023), pp. 1448–1459. ISSN: 0169-2070. DOI: https://doi.org/10.1016/j.ijforecast.2022.07.003. URL: https://www.sciencedirect.com/science/article/pii/S0169207022001017.

[25] Sam Allen, David Ginsbourger, and Johanna Ziegel. *Evaluating forecasts for high-impact events using transformed kernel scores.* 2022. arXiv: 2202.12732.

[26] Sam Allen et al. "Incorporating the North Atlantic Oscillation into the post-processing of MOGREPS-G wind speed forecasts". In: *Quarterly Journal of the Royal Meteorological Society* 147.735 (2021), pp. 1403–1418. DOI: https://doi.org/10.1002/qj.3983. eprint: https://rmets.onlinelibrary.wiley.com/doi/pdf/10.1002/qj.3983. URL: https://rmets.onlinelibrary.wiley.com/doi/abs/10.1002/qj.3983.

[27] S. Baran and S. Lerch. "Mixture EMOS model for calibrating ensemble forecasts of wind speed". In: *Environmetrics* 27.2 (Jan. 2016), pp. 116–130. ISSN: 1099-095X. DOI: 10.1002/env.2380. URL: http://dx.doi.org/10.1002/env.2380.

[28] Sam Allen. *Weighted scoringRules: Emphasising Particular Outcomes when Evaluating Probabilistic Forecasts*. 2023. arXiv: 2305.07312 [stat.CO]. URL: https://arxiv.org/abs/2305.07312.

[29] Sam Allen et al. "Weighted Verification Tools to Evaluate Univariate and Multivariate Probabilistic Forecasts for High-Impact Weather Events". In: *Weather and Forecasting* 38.3 (2023), pp. 499–516. DOI: 10.1175/WAF-D-22-0161.1. URL: https://journals.ametsoc.org/view/journals/wefo/38/3/WAF-D-22-0161.1.xml.

[30] D.S. Wilks. *Statistical Methods in the Atmospheric Sciences*. International Geophysics v. 100. Elsevier Science, 2006. ISBN: 9780127519661. URL: https://books.google.nl/books?id=_vSwyt8_OGEC.

[31] Nigel Roberts et al. "IMPROVER: The New Probabilistic Postprocessing System at the Met Office". In: *Bulletin of the American Meteorological Society* 104.3 (2023), E680–E697. DOI: 10.1175/BAMS-D-21-0273.1. URL: https://journals.ametsoc.org/view/journals/bams/104/3/BAMS-D-21-0273.1.xml.

[32] Thordis L. Thorarinsdottir and Tilmann Gneiting. "Probabilistic forecasts of wind speed: ensemble model output statistics by using heteroscedastic censored regression". In: *Journal of the Royal Statistical Society: Series A (Statistics in Society)* 173.2 (2010), pp. 371–388. DOI: https://doi.org/10.1111/j.1467-985X.2009.00616.x. eprint: https://rss.onlinelibrary.wiley.com/doi/pdf/10.1111/j.1467-985X.2009.00616.x. URL: https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-985X.2009.00616.x.

[33] Laurens Haan and Ana Ferreira. *Extreme Value Theory: An Introduction*. Springer, Jan. 2006. ISBN: 978-0-387-23946-0. DOI: 10.1007/0-387-34471-3.

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

[35] Y. LeCun et al. "Backpropagation Applied to Handwritten Zip Code Recognition". In: *Neural Computation* 1.4 (1989), pp. 541–551. DOI: 10.1162/neco.1989.1.4.541.

[36] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[37] François Chollet et al. *Keras*. https://keras.io. 2015.

[38] Simon Hakvoort. *Improving Probabilistic Forecasting of Extreme Wind Speeds in the Netherlands*. Accessed: 2024-10-31. 2024. URL: https://github.com/SimonHakvoort/Improving-Probabilistic-Forecasting-of-Extreme-Wind-Speeds-in-the-Netherlands.

[39] Yoshihiko Ozaki et al. "Multiobjective tree-structured parzen estimator for computationally expensive optimization problems". In: *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*. GECCO '20. Cancún, Mexico: Association for Computing Machinery, 2020, pp. 533–541. ISBN: 9781450371285. DOI: 10.1145/3377930.3389817. URL: https://doi.org/10.1145/3377930.3389817.

[40] Takuya Akiba et al. "Optuna: A Next-generation Hyperparameter Optimization Framework". In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2019.

[41] Hajo Holzmann and Bernhard Klar. "Focusing on regions of interest in forecast evaluation". In: *The Annals of Applied Statistics* 11.4 (2017), pp. 2404–2431. DOI: 10.1214/17-AOAS1088. URL: https://doi.org/10.1214/17-AOAS1088.