A calibrated and consistent combination of probabilistic forecasts for the upcrossing of several precipitation thresholds using neural networks

P. Schaumann[1], R. Hess[2], M. Rempel[2], U. Blahak[2] and V. Schmidt[1]

[1]*Institute of Stochastics, Ulm University, Ulm, Germany*

[2]*Deutscher Wetterdienst, Offenbach, Germany*

*

ABSTRACT

The seamless combination of nowcasting and numerical weather prediction (NWP) aims to provide a functional basis for very-short-term forecasts, that are essential e.g. for weather warnings. In this paper we propose a statistical method for precipitation using neural networks (NN) that combines nowcasting data from DWD's radar based RadVOR system with postprocessed forecasts of the high resolving NWP ensemble COSMO-DE-EPS. The postprocessing is performed by Ensemble-MOS of DWD. Whereas the quality of the nowcasting projections of RadVOR is excellent at the beginning, it declines rapidly after about 2 hours. The postprocessed forecasts of COSMO-DE-EPS in contrast start with lower accuracy but provide meaningful information on longer forecast ranges. The combination of the two systems is performed for probabilities that the expected precipitation amounts exceed a series of predefined thresholds. The resulting probabilistic forecasts are calibrated and outperform both input systems in terms of accuracy for forecast ranges from 1 to 6 hours as shown by verification.

The proposed NN-model generalises a previous statistical model based on extended logistic regression, which was restricted to only one threshold of 0.1 mm. The various layers of the NN-model are related to the conventional design elements (e.g. triangular functions and interaction terms) of the previous model for easier insight.

1. Introduction

Accurate and reliable forecasts of precipitation in the very-short-term range up to 6 h (Wang et al. 2017) in terms of location and time are required in order to issue targeted warnings. Early warnings support to increase the lead time for decision makers in hydrological and emergency services and can help to diminish possible damages caused by floodings or debris flows. Commonly used bases of these warnings in current operational weather forecasting are nowcasting systems and numerical weather prediction (NWP). Both approaches can provide valuable warning guidances, however, for different forecast lead times.

Methods for nowcasting of precipitation usually rely on remote-sensing observations from a radar network. In a processing step, the obtained radar reflectivities are transformed to estimates of the current rainfall rate. Based on the Lagrangian persistence approach the latest rainfall rates are extrapolated in space and time using a previously determined motion vector field (Germann and Zawadzki 2002). The dynamical evolutions of the convective cells are not considered. Hence, due to the spatial dependency of the lifetime of precipitation fields (Venugopal et al. 1999), such forecasts are able to produce a significant skill as long as the Lagrangian persistence assumption is valid (Zawadzki et al. 1994). The predictability reaches from hours at scales of several hundred kilometers down to minutes when considering developing thunderstorms (Foresti and Seed 2014).

In contrast, NWP models explicitly simulate the physical evolution of precipitation fields. Forecast errors result from initial and boundary conditions and from approximating physical equations and their inexact solutions due to finite resolutions in time and space. Especially the simulation of cloud microphysics by sub-grid parameterizations is crucial for precipitation

forecasts (Nicolis et al. 2009). In Stephan et al. (2008), it is shown that deficiencies in the simulated rainfall intensities are attributed to shortcomings in the physics parameterizations.

In order to reduce the intrinsic uncertainties accompanying NWP forecasts, ensembles were introduced. The ensembles consist of several realizations of a model run, however, differ in initial conditions, boundary conditions and often physical parameterizations (Palmer 2002; Gebhardt et al. 2011). Ensemble forecasting provides users with information on the possible range of weather scenarios to be expected.

Nevertheless, multiple runnings of a NWP model can only reduce the random errors and not the aforementioned structural model deficiencies. Therefore, ensemble members are not able to represent the whole spectrum of uncertainties (Scheuerer 2014). Hence, a statistical postprocessing is necessary to reduce systematic biases and an often experienced underdispersive behavior of ensemble forecasts (Gneiting et al. 2005).

Even with a bias corrected and distributional improved NWP prediction, the NWP forecasts still exhibit various errors in shorter time and small spatial scales. Therefore, a statistical combination with extrapolated nowcasting can improve the skill. The so-called seamless combination aims to create a unique and consistent forecast regardless of location and lead time (Brunet et al. 2015).

Vannitsem et al. (2020) point out that the combination of nowcasting and NWP forecasts may basically divided whether they take place in physical or probability space. NIMROD (Golding 1998) as one of the first combination schemes is based on a simple lead-time-dependent weighting function, where the weighting is based on a long-term comparative verification of both initial forecast systems. In INCA (Haiden et al. 2011), the weight for NWP forecasts linearly increases from the beginning until it reaches 1 at a lead time of +4 h.

The Short-Term Ensemble Prediction System (STEPS), see Bowler et al. (2006) and Seed et al. (2013), represents a more advanced combination method. Herein, tendencies in the latest observations and the NWP skill are quantified in real time and used to adjust weights combining the nowcast extrapolation and the NWP forecast, in dependence on lead time and spatial scale. Additionally, a forecast ensemble is generated due to the replacement of non-predictable spatial scales with correlated stochastic noise. Due to the emerging of nowcasting ensembles, efforts were made to not only use the forecast skill as objective combination metric but also the ensemble spread. Recently, Nerini et al. (2019) utilize an ensemble Kalman filter to iteratively combine NWP forecasts with radar-based nowcasting extrapolations.

In reference to combination methods in probability space, the blending scheme of Kober et al. (2012) weights exceedance probabilities derived from output of the NWP model COSMO-DE-EPS with smoothed neighborhood probabilities computed from the deterministic nowcasting algorithm Rad-TRAM. The weights are based on a long-term verification. Combination of multi-model ensembles are carried out in Johnson and Wang (2012) and Bouttier and Marchal (2020).

In a previous study, Schaumann et al. (2020) propose the LTI-model as a modified logistic regression model for precipitation rates higher than $0.1\,\mathrm{mm\,h^{-1}}$. Additionally to the logistic regression, triangular functions and interaction terms are introduced to take the possible differences in the individual initial probabilistic forecasts into account and, furthermore, to increase the flexibility to compensate a possible underestimation and overestimation. The logistic regression model is a common tool in the area of probabilistic weather forecasting and has been used for the calibration of forecasts in ,e.g., Hamill et al. (2008).

The present study aims to generalize the LTI-model with the help of a neural network (NN). The network to be developed should not only satisfy the demands set for the LTI-model but should also provide consistent threshold exceedance probabilities, where consistency is understood such that the probabilities of exceeding a threshold monotonously decrease with increasing thresholds. This is not ensured when models are trained for each threshold independently. Further, the network should be able to represent forecast uncertainty with increasing lead time. For other extensions to the logistic regression model in order to ensure consistency, see Wilks (2009); Ben Bouallègue (2013).

As in Schaumann et al. (2020), the training data set is based on forecasts of RadVOR (Winterrath et al. 2012) and Ensemble-MOS (Hess 2020). RadVOR is a nowcasting system that provides deterministic extrapolation forecasts of radar-based rainfall estimates. Radar observations are obtained by the operational German radar network operated by Deutscher Wetterdienst (DWD). Exceedance probabilities from the deterministic extrapolation forecasts are derived by using the neighborhood approach described by Theis et al. (2005). Ensemble-MOS statistically postprocesses output from the ensemble of the convection-permitting COSMO-DE model, which was upgraded to COSMO-D2 on 15 May 2018. It provides probabilistic precipitation forecasts for various thresholds using logistic regression.

The present study is organized as follows. In Section 2, a brief overview of the utilized training data set is given. Section 3 provides a brief summary of the LTI-model. Afterwards, the development of a NN architecture for the model generalization regarding the simultaneous consideration of multiple thresholds is described in detail. Since NNs react sensitive on the chosen hyper-parameters, a hyper-parameter optimization approach is provided in the appendix, see Section A1. Results are stated in Section 4. Herein, sensitivities in the

choice of hyper-parameters are investigated and a combination example is given. Finally, in Section 5 conclusions are drawn and an outlook is given for possible future developments.

2. Data

As in Schaumann et al. (2020), we use forecasts of the DWD systems Ensemble-MOS and RadVOR as data sources. However, we now extend the considered training and forecast period by three months and have thus, altogether, 6 months of precipitation data from April to September 2016. Furthermore, in the previous study, we considered data for only one threshold ($0.1\,\mathrm{mm\,h^{-1}}$) whereas now we consider 9 precipitation thresholds $t_1 = 0.1$, $t_2 = 0.2$, $t_3 = 0.3$, $t_4 = 0.5$, $t_5 = 0.7$, $t_6 = 1$, $t_7 = 2$, $t_8 = 3$ and $t_9 = 5$, with $\mathrm{mm\,h^{-1}}$ as unit of measurement.

The data used from both sources for the results derived in the present paper span across Germany and parts of neighboring countries. In this section, we briefly recall the main properties of the forecast systems Ensemble-MOS and RadVOR as well as the calibrated rainfall estimates used as ground truth. For further details regarding this data set, we refer to our previous study.

a. Ensemble-MOS

The postprocessing system Ensemble-MOS of DWD is a model output statistics (MOS) system with the capability to statistically process ensemble forecasts resulting in hourly outputs of probabilistic forecasts. These forecasts are available on a regular grid of 20 20 km^2, for lead times up to +21 h, and for various weather elements to support weather warnings (Hess 2020). From the latter output variables, we consider the exceedance of nine precipitation thresholds. The Ensemble-MOS forecasts used in the present paper are

based on ensemble forecasts of DWD's 2016 operational convection-permitting NWP model COSMO-DE-EPS (Theis et al. 2012). In particular, the training of the Ensemble-MOS relies on COSMO-DE-EPS forecasts from 2011 to 2015.

b. RadVOR

Additionally to the Ensemble-MOS forecasts, we use data from the nowcasting method RadVOR (Weigl and Winterrath 2010; Winterrath et al. 2012). RadVOR provides every 5 minutes deterministic precipitation forecasts for lead times up to +2 h on a regular grid of size 1 1 km^2. These very-short term forecasts consist of two components. In a first step, estimates of the current rainfall rate are derived from radar reflectivities. Thereafter, these rainfall rates are then advected in 5 minute increments according to a previously estimated motion vector field. Note that the edges of the respective forecast domain are shifted accordingly.

For the combination with Ensemble-MOS using the model proposed in the present paper, the RadVOR forecasts are interpolated to the same grid and matching time intervals. For this, the 5-minute precipitation amounts are added up for a one-hour sum. Next, we consider the grid points with precipitation rates larger than the given threshold on the 1 1 km^2 grid. Finally, to estimate the probability of threshold exceedance, we compute the average exceedance on the 1 1 km^2 grid for a neighbourhood of each grid point on the 20 20 km^2 grid.

c. Calibrated hourly rainfall estimates

As a ground truth for the training and validation of our models, we use calibrated rainfall estimates on a 1 1 km^2 grid based on reflectivity measurements of DWD's operational radar

network. These rainfall estimates are adjusted by about 1,300 rain gauge measurements to reduce the error induced by the uncertainty of the relationship between radar reflectivities and precipitation amounts (Winterrath et al. 2012). Note that, in comparison to the previous study, the filter algorithm proposed by Winterrath and Rosenow (2007) for removing pixel artifacts has not been applied.

3. Neural Network Architectures

In Schaumann et al. (2020) we used a modified logistic regression model for the combination of two different probabilistic forecasts. This model is referred to as LTI-model in the following. Here, L stands for "logistic" while T and I refer to "triangular functions" and "interaction terms", respectively, which are the modifications of the standard logistic regression, where the latter one is called the L-model.

To begin with, we briefly summarize the basic idea of the LTI-model and, later on in Section 3b, we propose further modifications of it for the simultaneous consideration of multiple thresholds.

a. Modified logistic regression model

The introduction of the LTI-model aimed to develop a model for the seamless calibrated combination of the afore described forecast systems Ensemble-MOS and RadVOR, see Section 2. Here, statistical calibration refers to an ideal reliability diagram, which is a desirable property of probabilistic forecasts (Murphy and Winkler 1977, 1987). The combined forecast should outperform the individual initial forecasts for all relevant lead times in accuracy and reliability. For short lead times the extrapolated nowcasting of RadVOR outperforms the skill of Ensemble-MOS. However, with increasing lead time, forecast scores for RadVOR

drop rapidly in comparison to those for Ensemble-MOS, see Figure 1. Note that the green plots in Figure 1 illustrate the notable improvements achieved by the NN-model introduced in Section 3b below.

The LTI-model is based on the standard logistic regression model $f_L : [0;1]^n \to [0;1]$ for some $n > 1$, where n denotes the number of probabilistic input forecasts to be combined. From a mathematical point of view, the L-model estimates the conditional probability distribution of a dichotomous random variable $Y : \Omega \to \{0;1\}$, i.e. the occurrence of precipitation above a certain fixed threshold, conditioned on given realizations x_i of a family of random variables $X_i : \Omega \to [0;1]$ for $i \in \{1;\dots;n\}$, i.e. the probabilistic input forecast models. The random variables $Y; X_1;\dots; X_n$ are defined on some probability space $(\Omega; F; \mathsf{P})$, where Ω contains the vectors $(y; x_1;\dots; x_n)$ consisting of all possible forecasts $x_1;\dots; x_n$ of the n input forecast models $X_1;\dots; X_n$, and the precipitation occurrence indicator y. Thus,

$$f_L(x_1;\dots;x_n) \quad \mathsf{P}(Y = 1 \mid X_1 = x_1;\dots; X_n = x_n) \qquad \text{for all } x_1;\dots;x_n \in [0;1].$$

Note that in Schaumann et al. (2020) the special case $n = 2$ has been considered, where the probabilities $x_1; x_2$ originate from RadVOR and Ensemble-MOS, respectively. Then, the L-model combines these two input forecasts and provides calibrated forecast probabilities.

1) Triangular functions

For the case $n = 2$ the L-model has three weights $w_0; w_1; w_2 \in \mathsf{R}$ and is given by

$$f_{\mathrm{L}}(X_1; X_2) = \quad (w_0 + w_1 X_1 + w_2 X_2); \tag{1}$$

where $(x) = \frac{e^x}{e^x+1}$ is the logistic function. Note that there are individual weights for each forecast time step. Due to the small number of weights, the L-model is rather limited in how it combines the input forecast models X_1 and X_2. The weights merely allow for enough

flexibility to weight each forecast based on its overall forecast quality. However, the forecast quality of X_i might vary across the range of possible predictions within the interval $[0;1]$. This variation in forecast quality is expressed by the reliability diagram of X_i (see Figure 2 for examples) indicating for which values $x_i \in [0;1]$ the input forecast model X_i tends to over- or underestimate the occurrence of the event that $Y = 1$. Therefore, we proposed to choose the weights for each model $X_1; X_2$ in dependence of the forecast values x_1 and x_2. For this, a family of $m+1$ triangular functions $\tau_j : [0;1] \to [0;1]$ has been defined for some integer $m > 0$ and all $j = 0;\dots;m$ with

$$\tau_j(x) = \max\left\{0; 1 - \left|m\,x - \frac{j}{m}\right|\right\} \qquad \text{for all } x \in [0;1] \tag{2}$$

and $\sum_{j=0}^{m} \tau_j(x) = 1$ for $x \in [0;1]$. Thereby, for $i = 1;2$, the input forecast model X_i can be encoded as a random vector $\tau(X_i) = (\tau_0(X_i);\dots;\tau_m(X_i)) \in [0;1]^{m+1}$. Thus, at most two consecutive triangular functions $\tau_j(X_i)$ and $\tau_{j+1}(X_i)$ are non-zero, which is the case when the value of X_i falls between $\frac{j}{m}$ and $\frac{j+1}{m}$. Now, instead of the forecast models X_1 and X_2, we pass the random vectors $\tau(X_1)$ and $\tau(X_2)$ to the L-model and call that the LT-model. For some weights $w_{ij} \in \mathbb{R}$, the latter model is defined as

$$f_{\mathrm{LT}}(X_1; X_2) = f_{\mathrm{L}}(\tau(X_1); \tau(X_2)) = \sum_{i=1}^{2} \sum_{j=0}^{m} w_{ij}\,\tau_j(X_i) \;.$$

If we now consider a reliability diagram with $m+1$ bins, then each weight for a triangular function corresponds to one bin and, therefore, allows the model to produce a calibrated forecast. Note that the LT-model does not use a weight w_0 like in Eq. (1), sometimes called "bias" or "intercept", because w_0 is redundant as the triangular functions sum up to 1 for any $x \in [0;1]$.

2) Interaction terms

The weights in the L- and LT-models are chosen for either one of the two input forecast models $X_1; X_2$, irrespective of the other forecast model. However, it might be sensible to choose different weights, depending on whether both forecasts agree or disagree on the probability of occurrence of the event $Y = 1$. For this, we consider four additional predictors $_1(X_1; X_2); : : : ; \ _4(X_1; X_2)$, called interaction terms, where $_1(X_1; X_2) = \sqrt{X_1 X_2}$, $_2(X_1; X_2) = \sqrt{(1 \quad X_1)X_2}$, $_3(X_1; X_2) = \sqrt{X_1(1 \quad X_2)}$, $_4(X_1; X_2) = \sqrt{(1 \quad X_1)(1 \quad X_2)}$.

Like in the previous section, where we have considered the vectors $(X_i) = (\ _0(X_i); : : : ; \ _m(X_i))$ for $i = 1; 2$, we now apply triangular functions to the interaction terms and pass the random vectors $(\ _i(X_1; X_2)) = (\ _0(\ _i(X_1; X_2); : : : ; \ _m(\ _i(X_1; X_2)))$ for $i = 1; 2; 3; 4$ to the model, i.e.,

$$\begin{aligned} f_{\mathrm{LTI}}(X_1; X_2) &= f_{\mathrm{L}}(\ (X_1); \ (X_2); \ (\ _1(X_1; X_2)); : : : ; \ (\ _4(X_1; X_2))) \\ &= \sum_{i=1}^{2} \sum_{j=0}^{m} w_{ij} \ _j(X_i) + \sum_{i=1}^{4} \sum_{j=0}^{m} w'_{ij} \ _j(\ _i(X_1; X_2)) \ ; \end{aligned}$$

where $w_{ij}; w'_{ij} \ 2 \ \mathsf{R}$ are some weights.

3) Model training

For training the LTI-model, a rolling-origin with reoptimization scheme (Armstrong and Grohman 1972) is used in order to simulate the operational conditions. For this, the available data is not split up in separate training and test datasets, but it is split by a point in time , which represents the "present", into "past data" and "future data". In each step of the rolling-origin scheme the model is trained on "past data" and then validated based on predictions made for time steps ahead of , on which the model has not been trained yet. At

the end of each step, is moved forward in time by one time step. This process is repeated until traversed the entire dataset.

b. Generalization of the LTI-model for multiple thresholds, using neural networks

1) Overview

In the present paper, we propose a number of modifications of the LTI-model, which allow for the combination of forecasts for several thresholds by one common model. Note that the LTI-model, being a modified logistic regression model, can be seen as a specific type of a NN. Therefore, more general variants of NNs are a natural choice to make further extensions of the LTI-model.

In NNs two types of parameters are distinguished: hyper-parameters and trainable parameters. Hyper-parameters determine the architecture of the NN-model (e.g. the numbers and types of layers and neurons). They have to be determined before fitting the trainable parameters to a data set. The trainable parameters are the weights within each layer. The fitness of a specific architecture as defined by the hyper-parameters is highly dependent on the problem to be solved. While there are some guidelines on how to design a NN, it is impossible to tell how the choice of a hyper-parameter affects the performance before fitting and validating the model (Bergstra et al. 2011). For the optimization of the hyper-parameters of our NN-model, we propose an algorithm in Section A1.

In the following sections we give a brief overview of the components of the proposed NN-model. Each component is either a generalization of a part of the LTI-model or a newly added modeling component. For each of them we introduce a number of hyper-parameters, which define the architecture of the NN-model and which have to be determined before training of the model can begin. In total there are 10 hyper-parameters. Each hyper-

parameter has a set of possible configurations, which we denote by $H_i = \{c_i^1, \ldots, c_i^{m_i}\}$ for $i = 1, \ldots, 10$. Thus, the architecture of the NN-models considered in this paper is defined by a vector $h \in H_1 \times \ldots \times H_{10}$, which contains exactly one possible configuration for each hyper-parameter. For each $h \in H_1 \times \ldots \times H_{10}$, the NN-model consists of the following layers: (i) Zero to five convolutional layers, (ii) Zero or one dense layer for interaction terms (iii) Zero or one layer consisting of triangular functions, (iv) one softmax layer. For a schematic representation of the model architecture, see Figure 3. With the exception of convolutional layers, each layer in this NN architecture corresponds to a component of the LTI-model, whereby the dense layer and softmax layer are more general than their LTI-counterparts. For an introduction to deep learning and explanations regarding topics like "activation function" or "convolutional layer", see Chollet (2017).

2) Convolutional layers for model input

With increasing lead time, it becomes more difficult to make accurate predictions due to increased forecast uncertainty. In particular, this might cause forecast models to be imprecise in the prediction of the location of precipitation events. For probabilistic forecasts, this imprecision may manifest itself in two ways: (i) The precipitation events are predicted at wrong locations. Note that this is especially relevant for the RadVOR probabilities, which are based on a simple extrapolation and therefore do not take increased uncertainty for longer lead times into account. This results in equally sharp predictions for all lead times. (ii) The probability mass spreads out spatially. In other words, the forecast model predicts lower precipitation probabilities for a larger area to take the increased uncertainty into account.

In both cases, it is advantageous for a combination model to be aware of the predictions made at adjacent locations in order to combine two forecasts at a given location. For this, we consider convolutional layers (Zhang et al. 1990), which are commonly used for analyzing image data or data arranged on a regular grid. A NN with convolutional layers takes the information from a neighborhood of adjacent grid points into account, whereas the LTI-model combines forecasts point-wise, i.e., the model output for each grid cell depends only on the individual input forecasts at this location. The size of this neighborhood is determined by the sizes of the convolutional kernels, which map the neighborhood data onto a vector of a specified length.

In the present paper, all kernels are square-shaped and therefore the considered sizes refer to both height and width, e.g., a kernel size of 3 refers to a neighborhood of 3 3 grid cells. Both the size of the kernel and the length of its output are hyper-parameters of the layer. The input of a convolutional layer is a tensor $I\ 2\ \mathrm{R}^{b_x \times b_y \times b_z}$, where b_x and b_y define the size of the forecast grid in x- and y-direction, respectively. Furthermore, b_z depicts the number of probabilistic predictions of Ensemble-MOS and RadVOR for 9 thresholds each. Thus, $b_z = 18$.

For technical reasons, the NN requires input data given on a rectangular grid. In cases where some of the grid cells are undefined (i.e. where no forecast is available), the input data is restricted to the largest rectangle, which contains only valid values, i.e., is NaN-free. Since the forecasts of RadVOR are based on radar measurements that are extrapolated according to a motion vector field, the edges of the respective forecast domain shift in time. Thus, the area containing data of both RadVOR and Ensemble-MOS depends on the magnitude of the motion vector field and may decrease with lead time. Another limitation is the total convolutional size, which has to fit inside the well-defined rectangle.

The total convolutional size is given by the formula: $c_1(c_2 \quad 1)+1$ with c_1 *2* H_1 (number of convolutional layers), c_2 *2* H_2 (kernel size of each layer). Due to this, some combinations of H_1 and H_2 result in an oversized convolution and, therefore, only combinations with a total convolutional size less than or equal to 13 are used. The latter convolution corresponds to a neighborhood of 260 km.

The following hyper-parameters and configurations are considered: number of convolutional layers with $H_1 = f0; 1; 2; 3; 4; 5g$, kernel size with $H_2 = f1; 2; 3; 5; 6; 7; 9; 11; 13g$, length of the output vector, i.e., the number of convolution matrices used by the layer (Chollet 2017), with $H_3 = f1; 2; 4; 6; 8; 10; 12; 14; 16g$, activation functions with $H_4 = ff_{\text{elu}}; f_{\text{exp}}; f_{\text{lin}}; f_{\text{sigmoid}}; f_{\text{relu}}; f_{\text{tanh}}g$ and L_2-regularization strength with $H_5 = f0; 10^{-7}; 5 \quad 10^{-7}; 10^{-6}; 5 \quad 10^{-6}; 10^{-5}g$.

3) Dense layer for interaction terms

For the LTI-model, the interaction terms were chosen by hand prior to model fitting. In the framework of NNs, the functionality of the interaction terms can be achieved with a densely connected layer of neurons, see e.g. Chollet (2017). In comparison to the LTI-model, a dense layer has the advantage that the shapes of the interaction terms are mostly determined by trainable parameters. The hyper-parameters for this layer are: number of neurons with $H_6 = f0; 2; 4; 6; 8; 10; 12g$, activation functions with $H_7 = ff_{\text{elu}}; f_{\text{exp}}; f_{\text{lin}}; f_{\text{sigmoid}}; f_{\text{relu}}g$ and L_2-regularization strength with $H_8 = f0; 10^{-7}; 5 \quad 10^{-7}; 10^{-6}; 5 \quad 10^{-6}; 10^{-5}g$.

4) Layer for triangular functions

The role of this layer is the integration of triangular functions into the NN-model. For a definition of and the rationale behind triangular functions, see Section 3a. The

only hyper-parameter for this layer is the number of triangular functions with $H_9 = \{0; 2; 4; 6; 8; 10; 12; 14\}$. Thus, for each $c \in H_9 \setminus \{0\}$, this layer applies the triangular functions $\Delta_0, \ldots, \Delta_c$ to the output of each neuron of the previous dense layer. In our case, for each $c' \in H_6$, the previous layer returns a vector of scalars $(x_1, \ldots, x_{c'})$. Hence the triangular functions layer has the output $\Delta_0(x_1), \ldots, \Delta_c(x_1), \ldots, \Delta_0(x_{c'}), \ldots, \Delta_c(x_{c'})$ of size $(c+1)c'$. To our knowledge, such functions are not used in conventional NNs (a similar concept are radial basis function networks (Park and Sandberg 1991)), but they can be manually constructed with the help of Keras backend functions (Keras 2020).

5) Softmax layer to ensure consistent predictions for all thresholds

To obtain exceedance probabilities for all considered thresholds $t_1, \ldots, t_m$ with $0 < t_1 < \ldots < t_m$ by means of LTI-models, a separate LTI-model would have to be trained for each threshold. However, this does not guarantee that the probabilities are decreasing monotonously for increasing thresholds, since the separate LTI-models have no knowledge about each other. Hence, an extended model is needed, of which the output is a vector of monotonously decreasing probabilities for the exceedance of the considered thresholds $t_1, \ldots, t_m$. Additionally, the data available for one threshold might be useful for the combination of other thresholds, too, since each probability is a point on the discrete cumulative distribution function of the same event and, therefore, they are interlinked.

To ensure that the components of the vector of combined forecasts are decreasing monotonously, we train the neural network on a multi-label classification problem with a Softmax layer (Bridle 1990). This can be seen as a generalization of the logistic regression model, which has been utilized in the LTI-model. While the logistic regression model estimates the conditional probability distribution of a dichotomous random variable

$Y : \Omega \to \{0, 1\}$, the softmax layer allows for estimating the conditional probability distribution of a random variable $Y' : \Omega' \to \{0, \ldots, m\}$ where $m \geq 1$. In our case, Y' models the exceedance of the considered precipitation thresholds at a given location. For this, let $T : \Omega' \to [0, \infty)$ be the precipitation amount and let $C_i = \{T \in [t_i, t_{i+1})\}$ denote the event that T takes values between the thresholds t_i and t_{i+1}, for $i = 0, \ldots, m$. For this, we formally introduce two further thresholds t_0 and t_{m+1}, where $t_0 = 0$ and $t_{m+1} = \infty$. We then put $Y' = i$ if $T \in [t_i, t_{i+1})$, i.e., Y' indicates which of the events $C_0, \ldots, C_m$ is occurring.

For the family of pairwise disjoint events $\mathcal{C} = \{C_0, \ldots, C_m\}$, the NN learns to predict a conditional discrete probability distribution $P_{C|I} = \{\mathrm{P}(C_0 \mid I), \ldots, \mathrm{P}(C_m \mid I)\}$, where $\mathrm{P}(C_i \mid I)$ is the conditional probability for the occurrence of event C_i given the model input I. Then, for each $j \in \{1, \ldots, m\}$, the conditional probability of the event that $T \geq t_j$ can be computed by $\mathrm{P}(T \geq t_j \mid I) = 1 - \sum_{i=0}^{j-1} \mathrm{P}(C_i \mid I)$. Clearly, from this it follows by definition that $\mathrm{P}(T \geq t_j \mid I) \leq \mathrm{P}(T \geq t_k \mid I)$ if $t_j < t_k$, and therefore the predictions of the NN are consistent for all thresholds.

6) Optimizer for trainable parameters

Another difference between the LTI-model introduced in Schaumann et al. (2020) and the NN-model considered in the present paper is the choice of the optimizer. Note that the optimizer controls how the trainable parameters change during the model training. This has a large influence on the model fitness. For a more detailed introduction to the operating principle of optimizers, we refer to Chollet (2017).

In our previous paper, a stochastic gradient descent with a constant learning rate has been used as optimizer for the LTI-model. While this is sufficient for the (relatively small) number of parameters of the LTI-model, the NN-model considered in the present paper requires a

more sophisticated optimizer, due to weaknesses of the classical stochastic gradient descent. Depending on the network architecture and the training data set, gradients for some weights might "vanish" at some point in training, see Glorot et al. (2011). This means that parts of the NN might receive only small updates or a sparse number of updates leading to stagnation of the training process. More recent optimizers address this problem by various means, e.g., gradient descent with momentum (Sutskever et al. 2013) or adaptive learning rates. This ensues, first, to scale up small gradients back to a reasonable size or, second, to compensate for a sparse number of updates of a weight. In the present paper, five different optimizers are investigated. All of them are based on adaptive learning rates, which leads to a thenth hyper-parameter with $H_{10} = f$Adam*;* Adagrad*;* Adadelta*;* Adamax*;* Nadam$g$. For more details on how each optimizer works, see Kingma and Ba (2015), Duchi et al. (2011), Zeiler (2012), and Dozat (2016).

c. Training and Validation

In the previous section, several modifications of the LTI-model have been discussed. Each of them introduces one or more hyper-parameters, needing to be determined before the NN can be trained. For this, a hyper-parameter optimization algorithm is employed, which is explained in Section A1 in more detail.

To train the NN, the available data is split into several parts. One for each step of the training process: (i) Training of different model architectures for the hyper-parameter search, using data of the period from 2016-04-01 to 2016-05-31, (ii) fitness evaluation of model architectures for the hyper-parameter search, using data from 2016-06-01 to 2016-06-30, (iii) pick best architecture based on evaluation results, (iv) training of the best architecture

(without validation), using data from 2016-04-01 to 2016-06-31, and (v) rolling origin update and validation with best architecture, using data from 2016-07-01 to 2016-09-30.

This kind of training process ensures that the choice of the best performing architecture and its validation are based on two different time intervals, since this would lead to a skewed validation result otherwise.

4. Results

a. Influence of hyper-parameters

1) Optimizer

For the hyper-parameter optimization, about 18000 model architectures have been evaluated for each considered lead time. The choice of the optimizer $c\ 2\ H_{10}$ has, by far, the largest impact on model fitness (see Section A1a). The distribution of model fitness for each optimizer is depicted in Figure 4. Since Adam, Adamax and Nadam outperform Adagrad and Adadelta for almost all model configurations, we will focus on results only with regard to Adam, Adamax and Nadam in the following discussion.

2) Convolutional layers

Furthermore, the model fitness is highly affected by the number of convolutional layers (selected from the set H_1) and their kernel size (selected from H_2), see Figure 5, where the results of the hyper-parameter optimization are visualized. While the difference between individual configurations is less pronounced for lead times +1 h, larger total convolutional sizes perform better for longer lead times. Thus, in situations of increased forecast uncertainty (e.g. for longer lead times), an improved forecast skill is achieved when considering

more adjacent grid cells. This is in agreement with the ideas of Theis et al. (2005) and Schwartz and Sobash (2017).

The activation functions f_{elu}; f_{linear}; f_{relu} and $f_{\tanh}$ seem to perform similarly well when compared to each other. In contrast, the functions $f_{\text{exponential}}$ and f_{sigmoid} exhibit a much worse behavior, in particular for model architectures with many convolutional layers. As an exception, f_{sigmoid} does not show this behavior for the lead time +6 h and even performs best out of all considered activation functions.

Models with larger lengths of output vectors (selected from H_3) tend to perform better, however, the difference is clearly pronounced only up to a vector length of 4.

It should be noted that the number of convolutional layers (selected from H_1) and their kernel size (selected from H_2) affect the output size of the NN in two different ways: (i) As explained in Section 3b, input data passed to the NN needs to be rectangular-shaped and defined on each grid cell in order to enable the NN to learn and to make a prediction. Note that the passed data domain underlies a large variability, due to unregular boundaries of and occasionally missing data within the input forecasts. The total convolutional size determines the possible minimum edge length of the data domain. (ii) Furthermore, the total convolutional size determines the size of the input area which is mapped to an output value. For example, for a total convolutional size of 5, a model input I of size 15 30 18 is mapped to a model output of size 11 26 9.

Due to these effects, the amount of available data used for training and validation depends on the total convolutional size. This might be an explanation for the decreased fitness of the four configurations $(1; 13)$; $(4; 4)$; $(3; 5)$; $(2; 7)$ of H_1 H_2 with the largest total convolutional size of 13, since they have much less output to be trained and validated on, see Figure 5.

For a total convolutional size of 9 (used for lead times +1 h, +2 h, +4 h and +5 h, see Table 1), the passed data domain I consists, on average, of about 292 grid cells in the considered time period (July, August, September 2016). This results in 636126 data points in total. In comparison to this, for a total convolutional size of 11 (used for lead times +3 h and +6 h, see Table 1), the passed data domain consists, on average, of about 160 grid cells with 344431 data points in total.

3) Triangular functions and interaction terms

In Figure 6 the effect of triangular functions and the dense layer (interaction terms) on the model fitness is visualized. In general, one might expect that more neurons in the dense layer perform better than less. Thus, at first glance, it seems to be counter-intuitive that 2 neurons perform worse than no neurons at all. A likely explanation of this is that few neurons act as a bottleneck that restricts the amount of information the NN can pass through the dense layer, whereas for zero neurons the layer and, therefore, the bottleneck is removed. Regarding triangular functions, typically 3 to 9 of them perform best for all lead times. However, some of these configurations can perform worse for specific lead times, e.g., 11 triangular functions for +4 h, and 5 triangular functions for +5 h, see Figure 6. The results visualized in Figure 6 show that model fitness for lead time +1 h behaves differently in comparison to the model fitnesses for longer lead times. See also Figure 5, where this effect can be observed, too.

4) Remaining hyper-parameters

The remaining hyper-parameters are the activation functions (selected from H_7) for the dense layer and the L_2 regularization strengths for the convolutional layers (selected from

H_5) and the dense layer (selected from H_8). However, these parameters do not seem to affect the model fitness in any significant way.

b. Validation of chosen architectures for different lead times

The model fitness has been evaluated on data for the months July, August and September 2016. Within this period of time and the passed domain I (depending the total convolutional size), the considered precipitation thresholds were exceeded as described below by the numbers of upcrossings (and corresponding relative frequencies in parenthesis).

(i) For a total convolutional size of 9 we have: 39303 (6:18%) for 0:1 mm, 31284 (4:92%) for 0:2 mm, 26402 (4:15%) for 0:3 mm, 20300 (3:19%) for 0:5 mm, 16459 (2:59%) for 0:7 mm, 12677 (1:99%) for 1 mm, 6264 (0:98%) for 2 mm, 3494(0:55%) for 3 mm, and 1375 (0:22%) for 5 mm.

(ii) For a total convolutional size of 11 we have: 20420 (5:93%) for 0:1 mm, 16195 (4:70%) for 0:2 mm, 13561 (3:94%) for 0:3 mm, 10330 (3:00%) for 0:5 mm, 8358 (2:43%) for 0:7 mm, 6386 (1:85%) for 1 mm, 3046 (0:88%)for 2mm, 1652 (0:48%) for 3 mm, 677 (0:20%) for 5 mm.

For each lead time, the parameter configurations are shown in Table 1, which were chosen by the hyper-parameter optimization algorithm explained in Section A1. For each chosen architecture validation scores and reliability diagrams are shown in Figures 1 and 2. Note that for some hyper-parameters several configurations perform equally well, which leads to random fluctuations between the choices for different lead times.

It can be seen that the NN-model makes less biased and more calibrated predictions with a higher Brier skill score in comparison to both initial forecasts provided by Ensemble-MOS

and RadVOR, for all lead times and thresholds. Although the RadVOR forecasts are only provided up to +2 h, the combined forecasts have better scores up to +6 h. A similar effect has been observed in our previous paper regarding the scores of combined forecasts obtained by the LTI-model. For more details on the used validation scores, see Wilks (2006).

Similar to the previously considered LTI-model, the NN-model of the present paper has improved reliability diagrams in comparison to both initial forecasts, see Figure 2. As expected, the reliability of forecasting models decreases with increasing lead times and increasing thresholds. In order to keep the reliability diagrams calibrated, the NN learns to lower its predictions accordingly to not overestimate the occurrence of precipitation, which leads to shorter curves in the reliability diagram for longer lead times and higher thresholds in comparison to both initial input models.

To test if it is actually necessary to run the hyper-parameter optimization algorithm for each lead time, we trained the chosen architecture for +1 h on the other five lead times, too. A visual comparison between the validations scores given in Figure 1 and the results for the +1 h-architecture showed only slight fitness differences. However, the reliability diagrams seem to be less calibrated, see Figure 2, We therefore decided to use a separate network architecture for each lead time.

c. Combination example

In Figure 7, forecasting results obtained by the combination of input data from Ensemble-MOS and RadVOR are shown, as an example, for the hour 6-7 am of 2016-07-21 and for three lead times (+1 h, +2 h, +3 h). Due to a larger total convolutional size of the NN-model for +3h, the size of the output is smaller. For shorter lead times, the forecast of the

NN-model closely resembles the forecast of RadVOR, while for increasing lead times, the predictions become more smooth and more dependent on the Ensemble-MOS prediction.

5. Conclusions

a. Summary of results

In this paper we presented a hyper-parameter optimization algorithm and NN architectures for the combination of two probabilistic forecasts, where we consider several precipitation thresholds simultaneously. The architectures chosen by the hyper-parameter optimization algorithm show improvements for all considered validation scores across all thresholds, and calibrate the resulting probabilities. Like the previously developed LTI-model, the NN-model considered in the present paper improves forecast scores also for lead times longer than +2 h, although RadVOR forecasts were only provided up to +2 h.

When comparing forecast scores of the LTI-model and the NN-model, it seems that the reliability diagrams of the LTI-model are smoother than those of the NN-model. This might be due to the fact that in the training process of the NN the training data must be cropped down to a smaller rectangular area with valid forecast values, which discards data points for the NN-model that are still available to the LTI-model. Also, the NN must predict consistent exceedance probabilities for several thresholds, which is an additional constraint to satisfy. It might be necessary for the NN-model to sacrifice some reliability for one threshold, in order to make better predictions for all considered thresholds on average.

b. Outlook & possible next steps

According to the results of the hyper-parameter search performed in the present paper, some hyper-parameters seem to be much more important than others. Thus, it might

be possible to further improve the architecture by adapting the search space. Since the optimizer and the number and size of convolutional layers have the largest influence on model fitness, additional optimizers and convolutional layer combinations should be investigated. Thus, in a forthcoming paper, we will investigate the numerical stability of the hyper-parameter optimization algorithm and how the chosen architectures in Table 1 compare ,e.g., to the second best architecture for each lead time.

Due to the restriction that the input of the NN must be rectangular and free of missing values, it should be considered to generate valid values by interpolation at grid points with missing values, or to pass an mask to the NN as additional input in order to specify, which values are valid. This would allow training without cropping of the data and also increase the area for which predictions can be made.

Furthermore, it would be interesting to investigate how additional information might affect the quality of combination, e.g., by increasing the resolution of the grid, by passing ensemble members directly to the NN without aggregation to probabilities, by adding an orography map to the input, or by including additional meteorological indicators. Given that a new dataset contains enough precipitation events for higher thresholds, the list of considered thresholds could be expanded.

APPENDIX

A1. Hyper-parameter optimization

To choose hyper-parameters by means of a systematic approach, various optimization algorithms have been developed, attempting to find correlations between the hyper-parameters of a model and its fitness by evaluating a number of different network architectures. The following problems arise in such algorithms. (i) Curse of dimensionality: For each additional hyper-parameter, the size of the search space grows exponentially. (ii) Training time: Depending on the size of the model, the size of the training dataset, and the available hardware, the evaluation of a network archtecture might take a considerable amount of computation time. (iii) Interactions between hyper-parameters: It is not enough to consider each hyper-parameter separately, because the best choice for some hyper-parameter might depend on the chosen configurations of other hyper-parameters. (iv) Non-deterministic model fitness: The fitting of a NN is a non-deterministic process and the weights of a model might not converge to the same optimum in repeated runs. This means that a single evaluation of a network architecture might not reflect the actual fitness of the architecture in general. For an introduction to hyper-parameter optimization in general and the concepts mentioned above in particular, see Hutter et al. (2019). To our knowledge, the following algorithm has not been proposed before.

To find a satisfactory model architecture despite the problems listed above, the proposed algorithm works according to the principle of Exploration & Exploitation, which is also explained in Hutter et al. (2019): At the beginning of the search, architectures across the whole search space are evaluated. With an increasing number of evaluations, promising candidates are prioritized.

In the following, we consider the search space $H = H_1 \times \cdots \times H_n$ being the Cartesian product of a family of domains $H_1, \ldots, H_n$ of n hyper-parameters for some integer $n \geq 1$. The set H_i consists of $m_i \geq 1$ available configurations of the i-th hyper-parameter, i.e., $H_i = \{c_i^1, \ldots, c_i^{m_i}\}$ for each $i = 1, \ldots, n$.

a. Fitness of an evaluation

In each iteration of the hyper-parameter optimization algorithm, the fitness $f(h)$ of a model architecture specified by $h = (c_1, \ldots, c_n) \in H$ is evaluated, where $c_i \in H_i$ for each $i = 1, \ldots, n$. The model architectures considered in this paper were trained for 6 epochs on a training data set (April + May 2016) and validated after each epoch on a separate validation data set (June 2016). Note that an epoch refers to one pass-through of the training dataset in the training process. We define the fitness $f(h)$ of a configuration $h \in H$ as the smallest model error achieved in any of the 6 epochs, whereas the model error is determined by the loss function (Chollet 2017) of the NN. Since we consider a classification problem in this paper, the loss function "categorical cross entropy" is used (Alla and Adari 2019).

To find out which number of epochs is sufficient, the model errors for 20 epochs have been determined for a number of model architectures. However, for most model architectures the minimum model error converged within the first 5 epochs. Therefore, we considered 6 epochs in order to derive the results obtained in this paper.

b. Selection of new hyper-parameter configurations

A common strategy to pick model configurations for evaluation is the so-called random search method, where a certain probability distribution, e.g. the uniform distribution, is

considered on the search space H from which new model architectures are sampled (Hutter et al. 2019).

The idea of the algorithm presented in this section is to start with a random search. However, after having made a number of evaluations, we can already estimate how the configuration $c_i \in H_i$ of a single hyper-parameter, for some $i \in \{1,\ldots,n\}$, affects the fitness of the model architecture $h = (c_1,\ldots,c_n)$. Based on this information, the probability distribution on H, from which further model architectures are sampled, can be adapted to favor model architectures which are more likely to perform well. With an increasing number of evaluations, the same concept can be applied to an increasing number of j hyper-parameters, where $j \in \{2,\ldots,n\}$, in order to find out which configurations $h' \in H'_J = H_{i_1} \times \ldots \times H_{i_j}$ for some subset $J = \{i_1,\ldots,i_j\} \subset \{1,\ldots,n\}$ perform well in combination with each other. In the following, we sometimes write $H'_{h'}$ instead of H'_J, in cases where we want to emphasize that a specific partial architecture h' belongs to the set H'_J. Furthermore, let $H^* = \bigcup_{J \in \mathcal{P}(\{1,\ldots,n\})} H'_J$ denote the set of all (partial) model architectures.

1) Definitions

The $(k+1)$-th choice of the model configuration $h_{k+1} \in H$, which is to be evaluated next, is made based on the set of previously evaluated configurations $E = \{h_1,\ldots,h_k\}$, for which the fitnesses $f(h_i);\ i = 1,\ldots,k;$ have been determined as described in Section 4a. Let $h' \in H'_J$ be a partial model architecture for a subset of hyper-parameters with indices $J = \{i_1,\ldots,i_j\} \subset \{1,\ldots,n\}$. Then define

$$E_{h'} = \{h \in E : h' \subset h\}$$

as the set of all evaluated hyper-parameter configurations h that share the same configurations with the partial architecture h'. For a given integer $s \in \mathbb{N} = \{1, 2, \ldots\}$ we define

$$E_s = \{h' \in H^* : |E_{h'}| \quad s; |E_{h' \cup \{c_i\}}| < s \text{ for all } c_i \in H_i; i \in \{1, \ldots, n\}\}$$

which is the set of partial architectures h' with at least s evaluations and for which all extensions $h' \cup \{c_i\}$ have less evaluations than s. In other words, E_s contains the largest partial architectures for which a minimum number s of evaluations exist.

When considering a partial architecture $h' \in H'_j$ for evaluation, we are not only interested whether E contains enough data to estimate $f(h')$, but also if all partial architectures in $H'_{h'}$ have enough evaluations. Hence we define

$$E_s^p = \{h' \in E_s : |E_{h'}| \quad \min_{g' \in H'_{h'}} |E_{g'}|^p\}$$

for a given value $p \in [1; 1)$, i.e., E_s^p is a subset of E_s containing all partial architectures h' with a number of evaluations $|E_{h'}|$ below an upper bound, which depends on the partial architecture $g' \in H'_{h'}$ with the smallest number of evaluations $|E_{g'}|$.

Furthermore, we define the median fitness $M_E(h')$ for a partial model architecture h' as

$$M_E(h') = \text{median}(\{f(h) : h \in E_{h'}\});$$

and, finally, the set $\Delta(h')$ of partial architectures g', which share $|h'|$ configurations with h' as

$$\Delta(h') = \{g' \in H : |h' \setminus g'| = |h'| \quad g \quad \text{for} \quad \in \mathbb{N}:$$

2) The algorithm

In this section we explain how we determine an initial partial architecture h'_0 and how we pick subsequent partial architectures $h'_1; \ldots; h'_k$ until their union is a full architecture that can be evaluated next and added to the set E.

For given values of s and p, we sample from the set of partial architectures E_s^p. Initially, the set E_s^p is empty, because E is empty since no architectures have been evaluated, yet. Note that E_s^p can also be empty due to the upper bound determined by the parameter p. In these cases, a partial architecture $h'_0 = fc_ig$ with a random configuration $c_i \ 2 \ H_i$ for a random hyper-parameter H_i is chosen. Otherwise, we sample h'_0 from the set E_s^p with probability $P_{E_s^p}$, where P_F is defined as

$$P_F(h') = \frac{2^{-M_E(h^0)=d}}{\sum_{g^0 \in F} 2^{-M_E(g^0)=d}}$$

for any partial architecture $h' \ 2 \ F$, a given set F of partial architectures and some $d > 0$, which is another parameter of the algorithm, along with s and p. Note that P_F is defined such that a partial architecture h' is twice as likely to be chosen than g' if $M_E(h') = M_E(g') + d$. Furthermore, once enough evaluations have been made such that E_s^p is non-empty, the algorithm will always pick a partial architecture from E_s^p. Without the upper bound, the first few partial architectures in E_s would be sampled ad infinitum, while other partial architectures, which have not been sampled often enough yet, would not be sampled at all. Therefore it is necessary to include the upper bound in E_s^p for the number of evaluations.

We now have the first part h'_0 of the architecture h, which we want to evaluate. Next, we successively determine more parts $h'_1; : : : ; h'_k$ until their union is a complete architecture $h = h'_0 \ [\ : : : \ [\ h'_k \ 2 \ H$. For this, let $\bar{h}'_j = h'_0 \ [\ : : : \ [\ h'_j$ be the union of all partial architectures up to h'_j.

For $j \ 2 \ f1; : : : ; kg$ we iteratively sample h'_j from $E_s^p \setminus \Delta(\bar{h}'_{j-1})$ with probability $P_{E_s^p \cap \Delta(\bar{h}'_{j-1})}(h'_j)$ with the smallest $\ 2 \ \mathbb{N}$ for which $E_s^p \setminus \Delta(h')$ is not empty. In other words, we choose h'_j such that it has at least one new configuration and also the largest possible overlap with $\bar{h}'_{j-1}$, and that the new configurations are likely to perform well in

combination with the previously chosen configurations. If $E_s^p \setminus \Delta(\bar{h}'_{j-1})$ is empty for all $\in \mathbb{N}$, the partial configuration $h'_j = \{c_i\}$ consists of a single randomly chosen configuration $c_i \in H_i$, where H_i is a random hyper-parameter, for which it holds that $H_i \setminus \bar{h}'_{j-1} = ;$.

Once enough architectures have been evaluated and we want to pick the best architecture based on E, we follow the same steps as described above, but instead of sampling partial architectures from E_s^p with probability $P_{E_s^p}$, we pick the partial architectures $h' \in E_s$ with the lowest $M_E(h')$ instead.

References

Alla, S. and Adari, S. K. (2019). *Beginning Anomaly Detection Using Python-Based Deep Learning*. Springer. ISBN: 978-1484251775.

Armstrong, J. S. and Grohman, M. C. (1972). A comparative study of methods for long-range market forecasting. *Management Science*, 19:211–221.

Ben Bouallègue, Z. (2013). Calibrated short-range ensemble precipitation forecasts using extended logistic regression with interaction terms. *Weather and Forecasting*, 28:515–524.

Bergstra, J. S., Bardenet, R., Bengio, Y., and Kégl, B. (2011). Algorithms for hyper-parameter optimization. In Shawe-Taylor, J., Zemel, R. S., Bartlett, P. L., Pereira, F., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems*, pages 2546–2554. Curran Associates, Inc.

Bouttier, F. and Marchal, H. (2020). Probabilistic thunderstorm forecasting by blending multiple ensembles. *Tellus A*, 72:1–19.

Bowler, N. E., Pierce, C. E., and Seed, A. W. (2006). STEPS: A probabilistic precipitation forecasting scheme which merges an extrapolation nowcast with downscaled NWP. *Quarterly Journal of the Royal Meteorological Society*, 132:2127–2155.

Bridle, J. S. (1990). Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In Soulié, F. F. and Hérault, J., editors, *Neurocomputing*, pages 227–236. Springer.

Brunet, G., Jones, S., and Ruti, P. M. (2015). *Seamless Prediction of the Earth System: from Minutes to Months*. World Meteorological Organization. ISBN: 978-9263111562.

Chollet, F. (2017). *Deep Learning with Python*. Manning Publications. ISBN: 978-1617294433.

Dozat, T. (2016). Incorporating nesterov momentum into adam. In *International Conference on Learning Representations, ICLR 2016, Workshop Track*. https://openreview.net/pdf/OM0jvwB8jIp57ZJjtNEZ.pdf.

Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159.

Foresti, L. and Seed, A. (2014). The effect of flow and orography on the spatial distribution of the very short-term predictability of rainfall from composite radar images. *Hydrology and Earth System Sciences*, 18:4671.

Gebhardt, C., Theis, S., Paulat, M., and Bouallègue, Z. B. (2011). Uncertainties in cosmo-de precipitation forecasts introduced by model perturbations and variation of lateral boundaries. *Atmospheric Research*, 100:168–177.

669 Germann, U. and Zawadzki, I. (2002). Scale-dependence of the predictability of precipitation
670 from continental radar images. Part I: Description of the methodology. *Monthly Weather*
671 *Review*, 130:2859–2873.

672 Glorot, X., Bordes, A., and Bengio, Y. (2011). Deep sparse rectifier neural networks. In Gor-
673 don G., Dunson D. and Dudík M., editors, *Proceedings of the Fourteenth International*
674 *Conference on Artificial Intelligence and Statistics, JMLR Workshop and Conference Pro-*
675 *ceedings*. pages 15:315–323.

676 Gneiting, T., Raftery, A. E., Westveld III, A. H., and Goldman, T. (2005). Calibrated
677 probabilistic forecasting using ensemble model output statistics and minimum CRPS es-
678 timation. *Monthly Weather Review*, 133:1098–1118.

679 Golding, B. (1998). Nimrod: A system for generating automated very short range forecasts.
680 *Meteorological Applications*, 5:1–16.

681 Haiden, T., Kann, A., Wittmann, C., Pistotnik, G., Bica, B., and Gruber, C. (2011). The
682 Integrated Nowcasting through Comprehensive Analysis (INCA) system and its validation
683 over the Eastern Alpine region. *Weather and Forecasting*, 26:166–183.

684 Hamill, T. M., Hagedorn, R., and Whitaker, J. S. (2008). Probabilistic forecast calibration
685 using ECMWF and GFS ensemble reforecasts. Part II: Precipitation. *Monthly Weather*
686 *Review*, 136:2620–2632.

687 Hess, R. (2020). Statistical postprocessing of ensemble forecasts for severe weather at
688 Deutscher Wetterdienst. *Nonlinear Processes in Geophysics*. DOI: 10.5194/npg-2019-64
689 (to appear).

Hutter, F., Kotthoff, L., and Vanschoren, J., editors (2019). *Automated Machine Learning: Methods, Systems, Challenges*. Springer.

Johnson, A. and Wang, X. (2012). Verification and calibration of neighborhood and object-based probabilistic precipitation forecasts from a multimodel convection-allowing ensemble. *Monthly Weather Review*, 140:3054–3077.

Keras (2020). Keras API reference: backend utilities. `https://keras.io/api/utils/backend_utils/`. accessed on 2020.08.20.

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In Bengio, Y. and LeCun, Y., editors, *3rd International Conference on Learning Representations, ICLR 2015, Conference Track Proceedings*. http://arxiv.org/abs/1412.6980.

Kober, K., Craig, G., Keil, C., and Dörnbrack, A. (2012). Blending a probabilistic nowcasting method with a high-resolution numerical weather prediction ensemble for convective precipitation forecasts. *Quarterly Journal of the Royal Meteorological Society*, 138:755–768.

Murphy, A. H. and Winkler, R. L. (1977). Reliability of subjective probability forecasts of precipitation and temperature. *Journal of the Royal Statistical Society: Series C*, 26:41–47.

Murphy, A. H. and Winkler, R. L. (1987). A general framework for forecast verification. *Monthly Weather Review*, 115:1330–1338.

Nerini, D., Foresti, L., Leuenberger, D., Robert, S., and Germann, U. (2019). A reduced-space ensemble kalman filter approach for flow-dependent integration of radar extrapolation nowcasts and nwp precipitation ensembles. *Monthly Weather Review*, 147:987–1006.

712 Nicolis, C., Perdigao, R. A., and Vannitsem, S. (2009). Dynamics of prediction errors under
713 the combined effect of initial condition and model errors. *Journal of the Atmospheric*
714 *Sciences*, 66:766–778.

715 Palmer, T. N. (2002). The economic value of ensemble forecasts as a tool for risk assessment:
716 From days to decades. *Quarterly Journal of the Royal Meteorological Society*, 128:747–774.

717 Park, J. and Sandberg, I. W. (1991). Universal approximation using radial-basis-function
718 networks. *Neural Computation*, 3:246–257.

719 Schaumann, P., de Langlard, M., Hess, R., James, P., and Schmidt, V. (2020). A calibrated
720 combination of probabilistic precipitation forecasts to achieve a seamless transition from
721 nowcasting to very short-range forecasting. *Weather and Forecasting*, 35:773–791.

722 Scheuerer, M. (2014). Probabilistic quantitative precipitation forecasting using ensemble
723 model output statistics. *Quarterly Journal of the Royal Meteorological Society*, 140:1086–
724 1096.

725 Schwartz, C. S. and Sobash, R. A. (2017). Generating probabilistic forecasts from convection-
726 allowing ensembles using neighborhood approaches: A review and recommendations.
727 *Monthly Weather Review*, 145:3397–3418.

728 Seed, A. W., Pierce, C. E., and Norman, K. (2013). Formulation and evaluation of a scale
729 decomposition-based stochastic precipitation nowcast scheme. *Water Resources Research*,
730 49:6624–6641.

731 Stephan, K., Klink, S., and Schraff, C. (2008). Assimilation of radar-derived rain rates
732 into the convective-scale model COSMO-DE at DWD. *Quarterly Journal of the Royal*
733 *Meteorological Society*, 134:1315–1326.

Sutskever, I., Martens, J., Dahl, G., and Hinton, G. (2013). On the importance of initialization and momentum in deep learning. Proceedings of the 30th International Conference on Machine Learning, Proceedings of Machine Learning Research (PMLR), pages 28:1139–1147.

Theis, S., Gebhardt, C., and Bouallegue, Z. B. (2012). *Beschreibung des COSMO-DE-EPS und seiner Ausgabe in die Datenbanken des DWD*. Deutscher Wetterdienst. https://www.dwd.de/SharedDocs/downloads/DE/modelldokumentationen/nwv/cosmo_de_eps/cosmo_de_eps_dbbeschr_201208.pdf.

Theis, S., Hense, A., and Damrath, U. (2005). Probabilistic precipitation forecasts from a deterministic model: A pragmatic approach. *Meteorological Applications*, 12:257–268.

Vannitsem, S., Bremnes, J. B., Demaeyer, J., Evans, G. R., Flowerdew, J., Hemri, S., Lerch, S., Roberts, N., Theis, S., Atencia, A., et al. (2020). Statistical postprocessing for weather forecasts–review, challenges and avenues in a big data world. Preprint: *arXiv:2004.06582*.

Venugopal, V., Foufoula-Georgiou, E., and Sapozhnikov, V. (1999). Evidence of dynamic scaling in space-time rainfall. *Journal of Geophysical Research*, 104:31599–31610.

Wang, Y., Coning, E., Harou, A., Jacobs, W., Joe, P., Nikitina, L., Roberts, R., Wang, J., and Wilson, J. (2017). Guidelines for nowcasting techniques. *WMO Publications*. published online: https://library.wmo.int/doc_num.php?explnum_id=3795.

Weigl, E. and Winterrath, T. (2010). Radargestützte Niederschlagsanalyse und -vorhersage (RADOLAN, RADVOR-OP). *Promet*, 35:78–86.

Wilks, D. S. (2006). *Statistical Methods in the Atmospheric Sciences*, volume 91. Academic Press.

Wilks, D. S. (2009). Extending logistic regression to provide full-probability-distribution mos forecasts. *Meteorological Applications*, 16:361–368.

Winterrath, T. and Rosenow, W. (2007). A new module for the tracking of radar-derived precipitation with model-derived winds. *Advances in Geosciences*, 10:77–83.

Winterrath, T., Rosenow, W., and Weigl, E. (2012). On the DWD quantitative precipitation analysis and nowcasting system for real-time application in German flood risk management. In Moore, R. J., Cole, S. J., and Illingworth, A. J., editors, *Weather Radar and Hydrology*, IAHS Proceedings and Reports, pages 351:323–329.

Zawadzki, I., Morneau, J., and Laprise, R. (1994). Predictability of precipitation patterns: An operational approach. *Journal of Applied Meteorology*, 33:1562–1571.

Zeiler, M. D. (2012). Adadelta: an adaptive learning rate method. Preprint: *arXiv:1212.5701*.

Zhang, W., Itoh, K., Tanida, J., and Ichioka, Y. (1990). Parallel distributed processing model with local space-invariant interconnections and its optical architecture. *Applied Optics*, 29:4790–4797.

LIST OF TABLES

Lead time	Nr. of conv. layers	Kernel size	Conv. activation	Conv. reg.	Conv. Output length
$+1h$	4	3	elu	5e-07	14
$+2h$	4	3	elu	5e-06	6
$+3h$	5	3	relu	0	8
$+4h$	1	9	elu	0	4
$+5h$	1	9	elu	5e-06	12
$+6h$	2	6	sigmoid	0	4

Lead time	Nr. of Neurons in dense layer	Dense activation	Dense reg.	Nr. of triangular func.	Optimizer
$+1h$	12	sigmoid	0	9	Nadam
$+2h$	12	tanh	0	5	Nadam
$+3h$	12	exponential	1e-06	3	Adamax
$+4h$	6	tanh	0	5	Adamax
$+5h$	10	sigmoid	0	3	Adamax
$+6h$	10	relu	1e-05	5	Adamax

Table 1. Selected con gurations of hyper-parameters for di erent lead times

LIST OF FIGURES

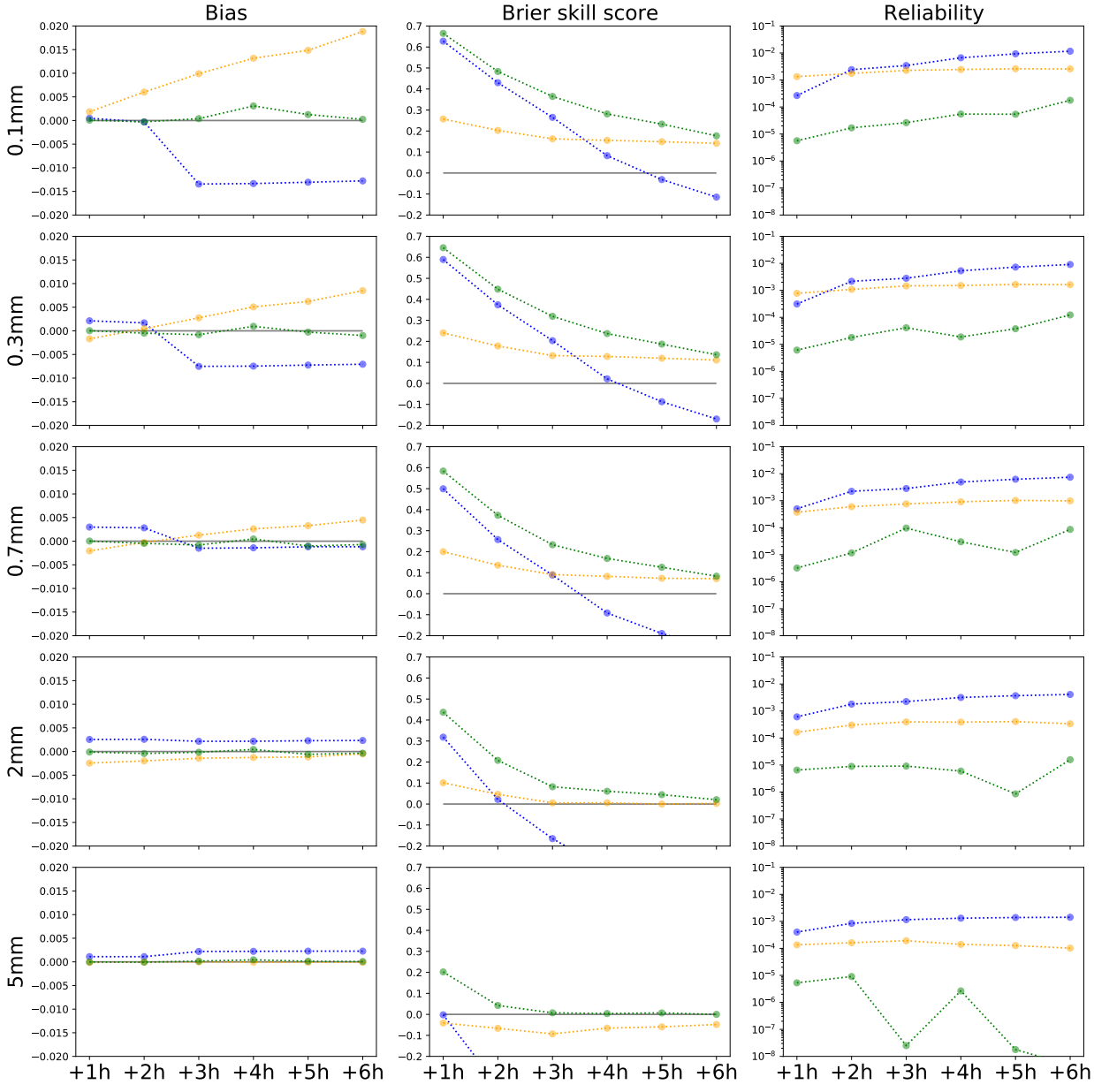

Fig. 1. Validation scores for 5 of the 9 considered thresholds for the NN-model (green), Ensemble-MOS (yellow) and RadVOR (blue). Left: bias, middle: Brier skill score, right: reliability.

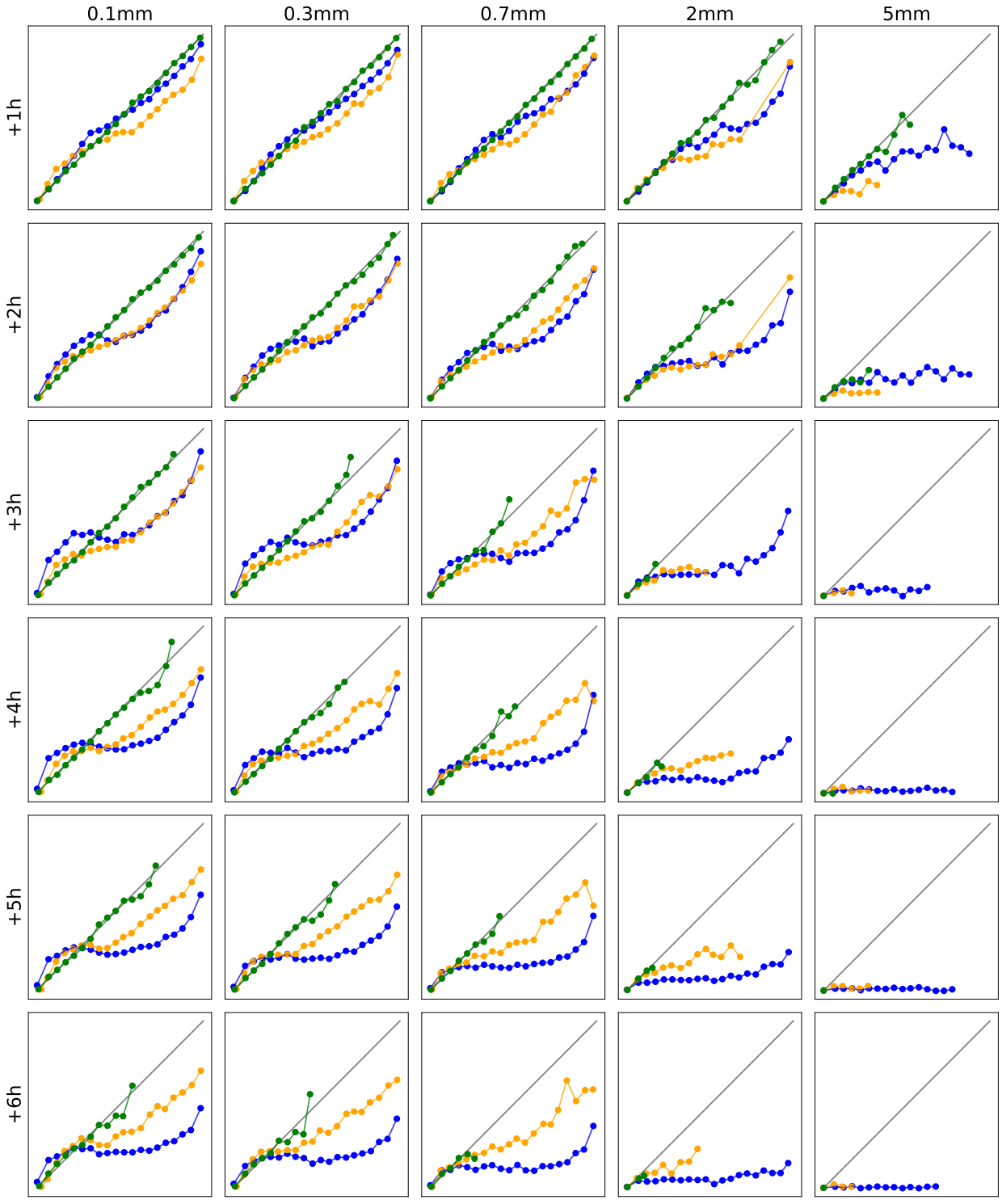

Fig. 2. Reliability diagrams for the NN-model (green), Ensemble-MOS (yellow) and RadVOR (blue) for 5 of the 9 considered thresholds (columns) and 6 lead times (rows). Bins with less than 100 exceedance probabilities are omitted.

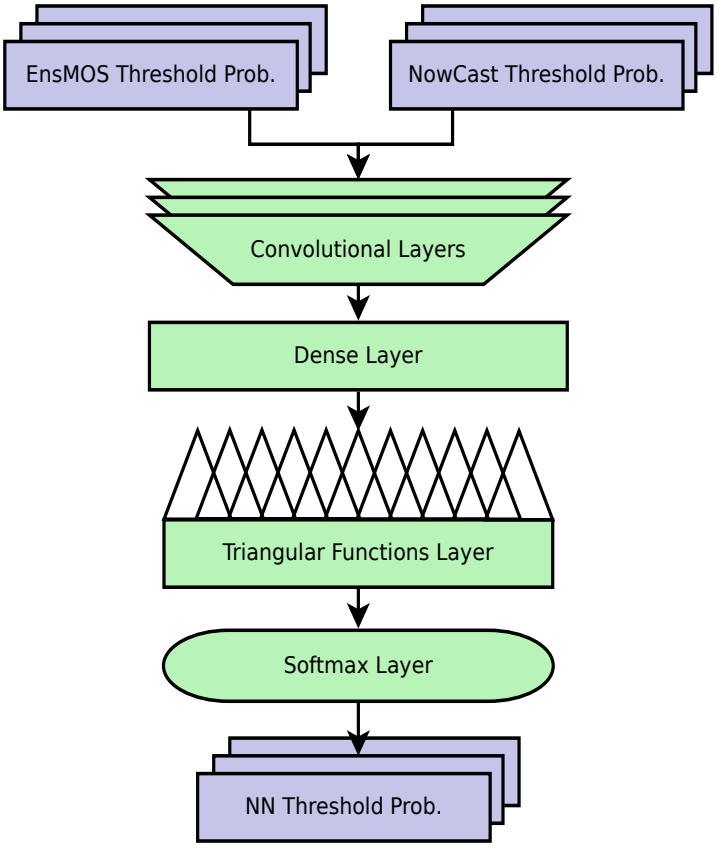

Fig. 3. A schematic representation of the network architecture (green) and the input and output data (blue). The arrows indicate the ow of information.

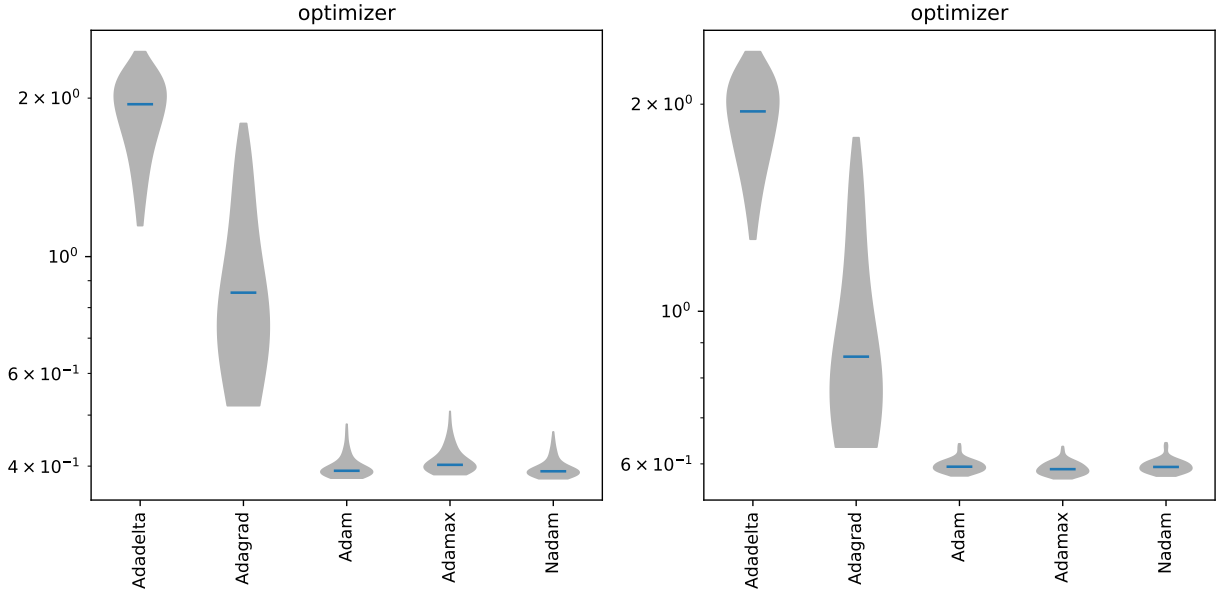

Fig. 4. Distribution of model fitness (categorical cross entropy) for the considered configurations of the first 9 hyper-parameters and the 5 optimizers, for lead time +1 h (left) and +6 h (right). The median fitness is depicted by a blue line. Due to very long tails, the distributions are only shown between the respective 5^{th} and 95^{th} percentiles.

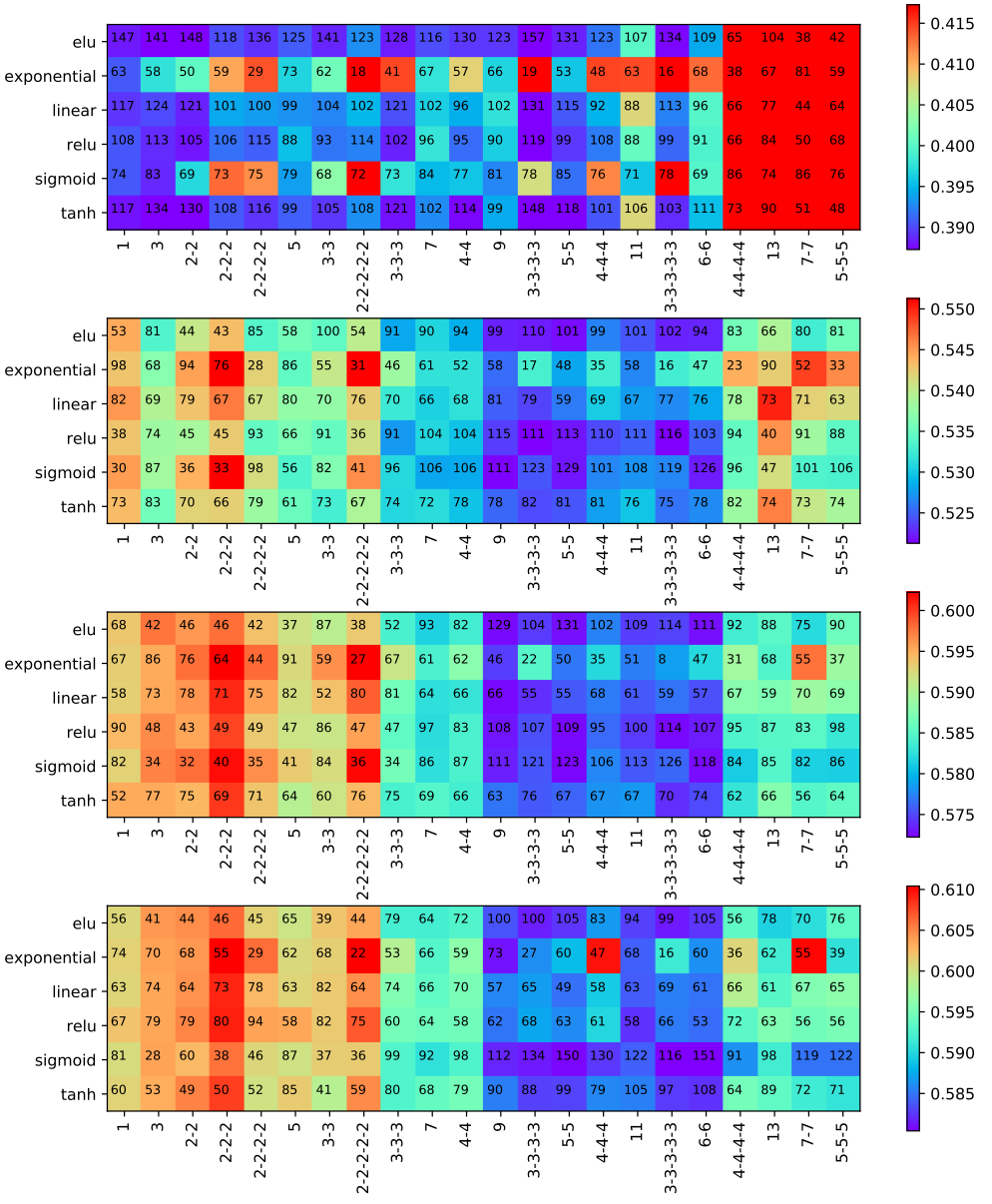

Fig. 5. Median model fitness (categorical cross entropy) for different hyper-parameter settings regarding the convolutional layers, and for several lead times: +1 h, +3 h, +5 h, +6 h (top to bottom). The color of each tile depicts the median model fitness in dependence on number and kernel size of the convolutional layers (*x*-axis) and activation function (*y*-axis). Whereby the *x*-axis labels correspond to the kernel size of each layer, e.g., 5 5 5 stands for three layers with a kernel size of five each. The number in each tile indicates the number of evaluations made by the hyper-parameter optimization algorithm.

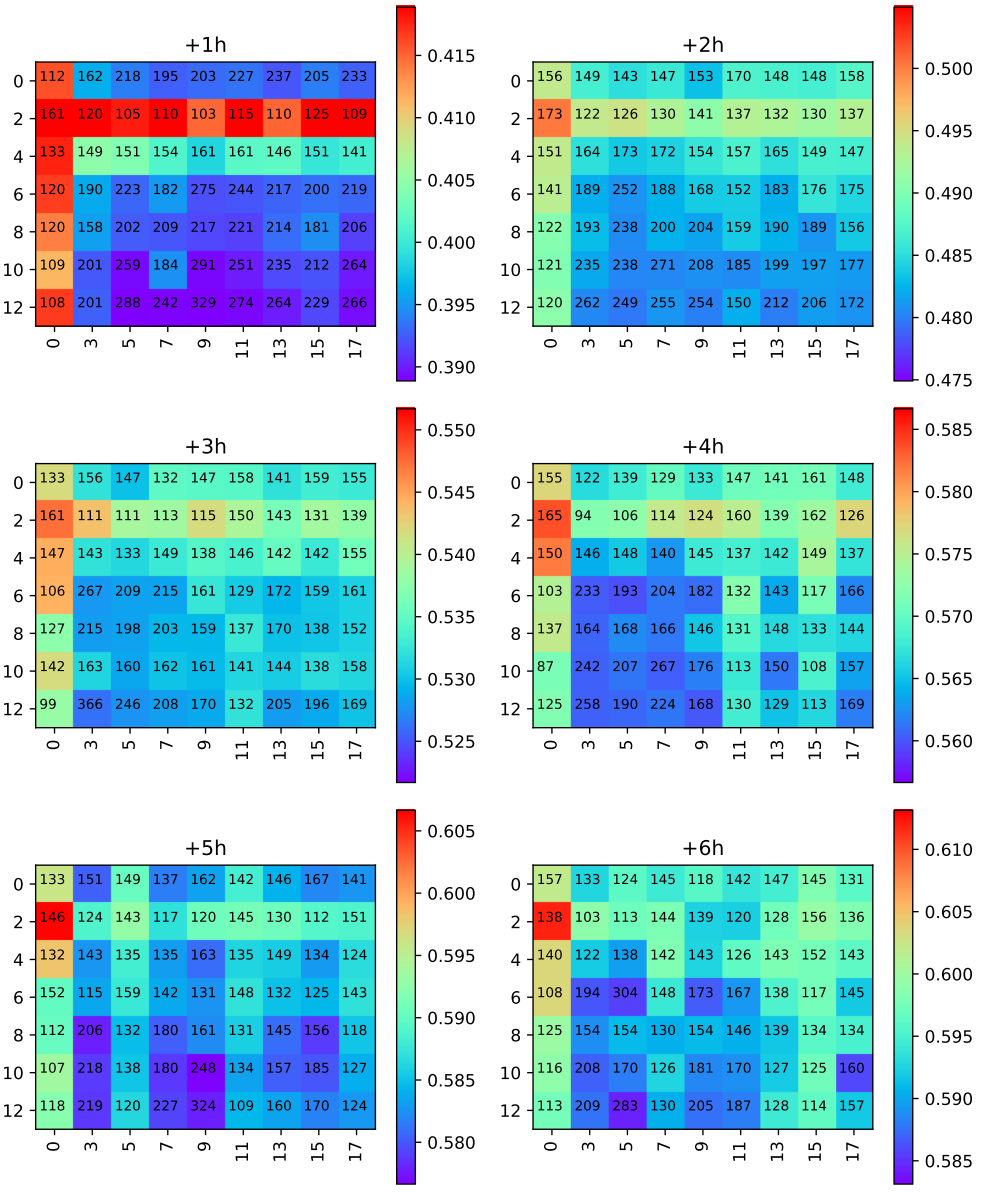

Fig. 6. Median model fitness (categorical cross entropy) for different hyper-parameter settings regarding triangular functions (x-axis) and interaction terms (y-axis), and for the lead times +1 h (top-left), +2 h (top-right), +3 h (middle-left), +4 h (middle-right), +5 h (bottom-left) and +6 h (bottom-right). The color depicts the median model fitness. The number in each tile indicates the number of evaluations made by the hyper-parameter optimization algorithm.