Machine Learning with Applications 2 (2020) 100007

Contents lists available at ScienceDirect

Machine Learning with Applications

journal homepage: www.elsevier.com/locate/mlwa

Short-term temperature forecasts using a convolutional neural network — An application to different weather stations in Germany

David Kreuzer [a], Michael Munz [a], Stephan Schlüter [b,*]

[a] Ulm University of Applied Sciences, Albert-Einstein-Allee 55, 89081 Ulm, Germany
[b] Ulm University of Applied Sciences, Prittwitzstraße 10, 89075 Ulm, Germany

ARTICLE INFO

Keywords:
Deep learning
LSTM
SARIMA
Temperature forecasts

ABSTRACT

Local temperature forecasts for horizons up to 24 h are required in many applications. A common method to generate such forecasts is the Seasonal Autoregressive Integrated Moving Average (SARIMA) model or, much simpler, the naïve forecast. In this paper, we test whether deep neural networks are able to improve on the results from the above mentioned methods. In addition to univariate long short-term memory (LSTM) networks, we present an alternative method based on a 2D-convolutional LSTM (convLSTM) network. For benchmarking our approach we set up a case study using data from five different weather stations in Germany. The SARIMA model and the univariate LSTM network perform quite well in the first few hours, but are then outperformed by the multivariate LSTM network and our convolutional LSTM network for longer forecast horizons. Besides, both multivariate approaches show better performance when the temperature is changing in the course of the day. Overall, our presented approach based on a convolutional LSTM network performs best on all used test data sets.

1. Introduction

Short-term temperature forecasts are required in many applications. Demand for such estimates has especially increased in the energy industry: With the rising share of renewable energy, one needs precise forecasts of its amount in order to maintain network stability. Forecasting models solar power, for example, include temperature values, which impact the solar modules' efficiency (Green, 2003). Besides, in electric power systems, having an accurate temperature forecasts is crucial to estimate energy consumption created e.g. by heating, ventilation, and air conditioning systems. Both wind and solar power are unstable energy sources which have caused a substantial increase of power plant redispatching in Germany during the past years. Thereby, redispatching activities include all kinds of short-term adjustments of individual power plant schedules. In 2017, for example, redispatching costs amounted to 423 million EUR. Besides, German grid operators paid 610 million EUR for curtailing renewable energy sources (and compensating the producers) in order to maintain grid stability (Bundesnetzagentur, 2017). Better temperature forecasts increase the precision of both the estimated wind and solar power and the estimated demand. Even small increases in forecasting precision can help to improve power plant dispatching and reduce costs. As the above mentioned numbers indicate the financial implications can be substantial. This is why, in this paper, we propose a new approach using concepts from data science. Since conventional weather models like the Lorenz model (Lorenz, 1963) are rather complex, data driven models are attracting more and more popularity. Among those approaches are deep learning concepts, i.e. neural networks with multiple hidden layers. Often, convolutional neural networks (CNNs) are used, where feature engineering is not needed. Instead, the feature extraction step is part of the networks layers and therefore is trained using data. Computational power for massive parallel computing (i.e. general purpose computation on graphics processing unit, short GPGPU) is increasing, hence more and more complex models are developed to predict various climate time series, among them also temperature. Neural networks are well suited for this task (Frnda et al., 2019). In Dong et al. (2018) or Xiaoyun et al. (2016), the authors already applied a recurrent neural network to predict temperature, wind speed or radiation, for example. Especially so-called long short-term memory

Abbreviations: ACF, Autocorrelation function; ARMA, Autoregressive moving average model; ARIMA, Autoregressive integrated moving average model; CNN, Convolutional neural network; convLSTM, Convolutional long short-term memory network; GoF, Goodness of fit; GPGPU, General purpose computation on graphics processing unit; LSTM, long short-term memory; MASE, Mean absolute scaled error; PACF, Partial autocorrelation function; RMSE, Root mean square error; RNN, Recurrent neural network; SARIMA, Seasonal autoregressive integrated moving average model; SN, Seasonal naive forecast

The code (and data) in this article has been certified as Reproducible by Code Ocean: (https://codeocean.com/). More information on the Reproducibility Badge Initiative is available at https://www.elsevier.com/physical-sciences-and-engineering/computer-science/journals.

* Corresponding author.
E-mail addresses: david.kreuzer@thu.de (D. Kreuzer), michael.munz@thu.de (M. Munz), stephan.schlueter@thu.de (S. Schlüter).

https://doi.org/10.1016/j.mlwa.2020.100007
Received 27 June 2020; Received in revised form 30 September 2020; Accepted 1 October 2020
Available online 1 November 2020

(LSTM) networks are often used because recurrent neural networks (RNNs) generally show good results when dealing with time series data (Lopez et al., 2016). Besides, LSTM cells in particular have the "ability to bridge very long time lags" (Hochreiter, 1997), which is crucial when dealing with seasonal data. In this work, we combine both concepts, i.e. convolutional and LSTM layers, to compute improved temperature forecasts. The combination of both showed good results in related applications (Kim & Cho, 2019; Liu et al., 2017; Xue et al., 2019). The focus of this paper is to show the applicability of deep learning for temperature forecasting based on multiple data channels of weather stations. To benchmark our deep neural network, we apply it to historic data from five different locations in Germany, namely Ulm, Kassel, Essen, Kempten, and Bremerhaven. The results are compared to classic methods of time series forecasting such as (seasonal) naive forecasts or the widely-used seasonal autoregressive integrated moving average (SARIMA) model. The SARIMA is chosen as we expect the temperature to show distinct daily seasonality. In addition, we compare our approach to a shallow neural network method, namely the LSTM network, both in an univariate and a multivariate version. In order to test our models we use weather data from multiple weather stations in Germany between 2009 and 2018, depending on their availability. Apart from air temperature, the following variables are used as inputs for all multivariate models: wind speed, wind direction, temperature, humidity, dew point temperature, air pressure, global radiation, and diffuse radiation, whereby we have a temporal granularity of one hour. We produce hourly forecasts up to 24 h in advance, whereby the first six of those hours are often referred to as nowcasting horizon in meteorology. As performance measures we use the root mean square error (RMSE) and the mean absolute scaled error (MASE). The RMSE is well known, fairly simple, and widely applied, which makes it easy to understand and interpret the results. The MASE, again, benchmarks a model's performance to the naive forecast. Hence, we immediately see whether the additional efforts of fitting a model to a data set pay off. Both methods together allow a reliable evaluation of a model's performance.

All models show good results for consecutive days with similar patterns. Results are different when the weather changes substantially in comparison to the day before, e.g. in case of a temperature drop. In this situation, both the multivariate LSTM and the convolutional LSTM (convLSTM) network perform significantly better than the other methods. Thereby, overall, considering all data sets and all weather situations, the convolutional LSTM network shows the best results given our chosen performance measures. Nevertheless, feeding more information is not always beneficial — we see that the SARIMA model and the simple LSTM network often outperform our relatively complex model in the first two to three hours. However, for larger forecasting horizons, our convolutional LSTM network delivers more accurate results.

The paper is structured as follows: In Section 2 we give a brief overview over existing concepts and introduce the models tested in this paper with a focus on neural networks (Section 2.3). The models are then trained and calibrated to a practical data set in Section 3 where we also explain this process. The results of this empirical analysis are discussed in Section 4. Section 5 concludes the paper summing up the main findings.

2. Methodology

Due to substantial demand, there is a wide range of models for generating temperature forecasts. When dealing with weather/climate forecasting models, literature commonly discriminates between fundamentally driven models and econometric models. The latter ones are basically always to some extent autoregressive and apply statistical methods to produce point or interval estimates for future points in time. Among the fundamental models, the largest ones are constructed and maintained by national weather institutes like for example the Deutscher Wetterdienst in Germany or the Norwegian Meteorological Institute, which is responsible for the well-known website www.yr.no Fundamental models are highly complex and are in general based on a large number of parameters and input data sets. The parameter calibration has to be supervised and input data needs to be checked for quality, hence their operation requires a certain manpower for maintenance purposes. As a consequence, such models are rather impractical for business or smaller applications. One can either then purchase forecasts or resort to econometric methods. In Section 2.1 we give a short overview before describing our benchmark, the SARIMA model, in Section 2.2. A brief introduction to neural network-based forecasting is given in Section 2.3.

Throughout this paper, $X_t, t = 1, \dots, T$ denotes the temperature time series with X_T being the most recent observation. As we consider hourly forecasts, the time stamp is of an hourly granularity. Forecasted values are denoted by $\hat{X}_{t+\Delta t}$, i.e. a six hours forecast starting with the last observation would be $\hat{X}_{T+6}$.

2.1. Existing methods

Besides the ones mentioned in the introduction, there is a wide range of possible models. A very simple model-free approach is the seasonal naive forecast, i.e. the assumption that the daily pattern is persistent. In mathematical terms: $\hat{X}_{T+k} = X_{T+k-24}, \forall k \geq 0$. Examples for general forecasting techniques are the exponential smoothing method (McNeil et al., 2015) or the SARIMA model. A more advanced approach is to use the wavelet transform as e.g. discussed by Schlüter and Deuschle (2014), who generate point forecasts by splitting up a time series in different frequencies. Alternatively, Herwartz and Schlüter (2016) propose a method to generate directional forecasts also using the wavelet transform. Curceac et al. (2019), again, focus on climate data and combine a kernel-based regression model with the SARIMA model. They also give a brief but wide literature overview over potential models ranging from artificial neural networks or support vector machines to Markov chains. Graf et al. (2019) use a combination of wavelets and artificial neural networks to forecast river temperature. Hassani et al. (2018) focus on weather anomalies and rely on an ensemble of 12 different forecasting models. Möller and Groß (2016) and Kann et al. (2011) also apply the concept of ensemble forecasts. The idea of using deep neural networks, more precisely a LSTM network, for temperature forecasting has been tested by Zhang et al. (2017). A similar neural network structure is used in Abdel-Nasser and Mahmoud (2019) to forecast photovoltaic power. The combination of CNNs and LSTM networks are used in related domains. For example, Qin et al. (2019) use it to predict the concentration of fine particles in an urban environment, and Kim and Cho (2019) forecast electricity consumption. Thereby, quite a few authors like (Xue et al., 2019) use the SARIMA model as a benchmark.

2.2. Seasonal autoregressive moving average model

For definitions in this section, we mostly follow Box et al. (2016), who discuss various types of autoregressive processes, i.e. stochastic processes whose values are (to some extent) determined by the previous realizations. The basic version is the autoregressive process of order $p, p \in \mathbb{N}$, short AR(p): A process X_t is explained by the past p observations plus an error term ϵ_t, which follows a certain random distribution, i.e.

$$X_t = \sum_{k=1}^{p} \phi_k X_{t-k} + \epsilon_t. \tag{1}$$

Often, $\epsilon_t \sim \mathcal{N}(0, \sigma^2)$ with $\sigma > 0$ being the standard deviation of the Gaussian distribution. If also ϵ_t shows some dependence on the previous q values, we add some coefficients θ_q and yield the autoregressive moving average model of order p and q, short ARMA(p, q):

$$X_t - \sum_{k=1}^{p} \phi_k X_{t-k} = \epsilon_t - \sum_{j=1}^{q} \psi_j \epsilon_{t-j}. \tag{2}$$

Besides, let B with $B^k X_t = X_{t-k}$ denote the so-called backshift operator. Based on B we can rewrite the left-hand side of Eq. (2) as follows:

$$X_t - \sum_{k=1}^{p} \phi_k X_{t-k} = X_t - \sum_{k=1}^{p} \phi_k B^k X_t = X_t \left(1 - \phi_1 B - \cdots - \phi_p B^p\right). \quad (3)$$

The expression $\Phi_p(B) := 1-\phi_1 B-\cdots-\phi_p B^p$ is called lag polynomial. Analogously we can rewrite the right side of Eq. (2):

$$\epsilon_t - \sum_{j=1}^{q} \psi_j \epsilon_{t-j} = \epsilon_t - \sum_{j=1}^{q} \psi_j B^j \epsilon_t = \epsilon_t \left(1 - \psi_1 B - \cdots - \psi_p B^q\right) \quad (4)$$

using the lag polynomial $\Psi_q(B) := 1 - \psi_1 B - \cdots - \psi_p B^q$. In case of non-stationarity, i.e. if the model is not stable regarding mean and or variance (for a detailed introduction, see Box et al., 2016), we apply the model not to X_t itself, but try differencing, i.e. we consider $Y_t = X_t - X_{t-1} := \Delta X_t$. This procedure can be repeated more than once, e.g. not Y_t but $\Delta Y_t = \Delta^2 X_t$ is considered. This differencing is repeated until $\Delta^d X_t$, is stationary. Thereby, d is called order of integration. Making use of the lag polynomials and the Δ operator we define the autoregressive integrated moving average model, short ARIMA(p, d, q) model, as

$$\Phi_p(B)\Delta^d X_t = \Psi_q(B)\epsilon_t, \quad (5)$$

with p, q denoting the degrees of autoregression and the lag of the moving average component, respectively. Incorporating seasonality into the model is also done using the backshift operator, i.e. $B^s X_t = X_{t-s}$. Analogously to above we define $\Phi_{seasonal}(B^s) := 1 - \phi_{seasonal} B^s$ and $\Psi_{seasonal}(B^s) := 1 - \psi_{seasonal} B^s$ for a seasonal lag s. Mind that the seasonal component (and its moving average element) could have more than one lag. However, for the sake of simplicity we restrict both to one, which is sufficient in most cases. If seasonality causes non-stationarity, one can try differencing on the seasonal level, i.e. compute $\Delta_s X_t := X_t - X_{t-s}$ before checking if regular differencing is required as well. This seasonal differencing can also be repeated D times. Eventually, we sum up everything to the very general seasonal autoregressive integrated moving average model, short SARIMA$(p, d, q)\times(1, D, 1)_s$, with Δ_s^D indicating that we compute the D-times difference of the seasonal lag,

$$\Phi_p(B)\Phi_{seasonal}(B^s)\Delta^d \Delta_s^D X_t = \Psi_q(B)\Psi_{seasonal}(B^s)\epsilon_t. \quad (6)$$

2.3. Neural networks

The main factor underlying the success of a machine learning algorithm is to find relevant features and the appropriate model. Thereby, today, neural networks are a powerful solution in many applications when conventional machine learning models reach their limits. Especially CNNs introduced by Lecun et al. (1998) are a solid alternative to classic approaches. Here, because we are dealing with time series, the network has to incorporate the temporal dependencies in the data set. Hence, important temporal information should be carried along in time inside the neural networks. This can be achieved with RNNs. Thereby, LSTM networks, introduced by Hochreiter (1997) seem to be a promising approach, as LSTM networks solve the problem of exploding or vanishing gradients RNNs often deal with Hochreiter (1998).

2.3.1. Convolutional neural network structures

Neural networks can be used for pattern recognition, for classification tasks, for prediction or regression tasks, or also for forecasting problems, which is the focus of this paper. CNNs had a big impact in machine learning as those networks have shown great success in many different visual and oral tasks, like image (Krizhevsky et al., 2017; Szegedy et al., 2015) and video recognition (Tran et al., 2015) as well as natural language processing (Kim, 2014). Therefore, many researchers adopted the model to other domains (Heryadi & Warnars, 2017; Molano-Mazon et al., 2018). In recent years CNNs have been also applied to weather forecasting tasks (Zhou et al., 2019). Lee et al. (2020) showed that LSTM networks can be outperformed by convolutional networks. In conventional machine learning approaches, features have to be defined and extracted manually based on expert knowledge. This step is very costly and complex, particularly when considering correlations between features. In contrast, CNNs are directly applied to raw data and the feature extraction step is part of the training process: inside the convolutional layers, filters for feature extraction are trained based on the data. This methodology is called end-to-end learning, as we completely learn the full pipeline structure from input data to network output.

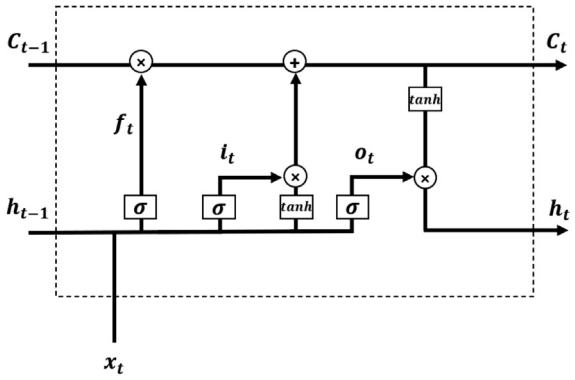


Fig. 1. Structure of a single LSTM cell, consists of an input gate i with activation vector i_t, an output gate o (activation vector o_t), and a forget gate f (activation vector f_t) (Alhirmizy & Qader, 2019).

As CNNs were primarily used to detect patterns in images, the input of a convolutional layer is expected to be matrix shaped. Multi-channel time series data of length N with C channels, as regarded in this paper, can therefore be either directly fed to the network as a one-dimensional data sequence of size $1 \times N$ with C network channels or as one image of size $C \times N$. In this paper, we use the second approach due to the fact that it is common for multi-channel time series data combined with LSTM networks, like for example in the work of Shi et al. (2015): Convolutional operations are applied to the data in time-domain as well as across the channels, which leads to calculation of feature maps based on multiple data input channels. Hereby the height of the image represents the current temporal excerpt, called window, of the data and the width of the image corresponds to the data channels. By applying a kernel $K \in \mathbb{R}^{\beta\times\gamma}$ to a matrix, new information can be gathered, where the kernel dimension γ determines how many adjacent features are incorporated. The convolution is defined by the $*$ operator at position (a, b) in the image I:

$$I_{filt}(a, b) = K * I(a, b) = \sum_{v=1}^{\beta} \sum_{v=1}^{\gamma} K(v, v) I(a - v, b - v). \quad (7)$$

The boundaries of the image must be treated separately, as neighbor pixels are missing, according to conventional filtering or convolution approaches, for example by using zero padding.

2.3.2. LSTM network structures

LSTM networks provide the opportunity to deal with long time sequences. This is done using so-called gates, which control the flow of information. Fig. 1 shows a LSTM cell consists of an input gate i with activation vector i_t, an output gate o (activation vector o_t), and a forget gate f (activation vector f_t). The input into the cell is x_t, which corresponds to a vector of observations at time t. In our case, since the LSTM network is coupled with the CNN, x_t is the output vector of the CNN at time t. The variable C_t indicates the cell state, i.e. the output of the cell at time t and h_t is the related hidden state. For the transition function σ, the Rectified Linear Unit (ReLU)- function is used:

$$\sigma(x) = \begin{cases} 0 & \text{for } x < 0, \\ x & \text{for } x \geq 0. \end{cases} \quad (8)$$

The names of the gates already give an indication of their tasks. The input gate updates the cell state with new information according to:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i),$$
$$C_t = tanh(W_C \cdot [h_{t-1}, x_t] + b_C). \quad (9)$$

The use of the tanh function is due to the fact that exploding gradients can be avoided and the function is continuously differentiable. The forget gate, again, allows to drop unnecessary information in order to prevent the cell state from growing indefinitely. The amount of information the network forgets can vary, depending on the hidden state of the previous layer h_{t-1} and the new input x_t:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f). \quad (10)$$

The output depends on the previous hidden state and the current cell state.

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o),$$
$$h_t = o_t \cdot tanh(C_t). \quad (11)$$

In accordance with the usual notation, W and b are weights and biases belonging to the gates.

For our new method we combine a CNN and a LSTM network to form a convLSTM network (Ordóñez & Roggen, 2016), described in detail in Section 3.2.

3. An empirical application – Temperature data from different cities in Germany

We compute hourly out-of-sample forecasts for the upcoming 24 h. SARIMA calibration and network training is done based on rolling time-windows of our data sets. Results are then compared using various goodness-of-fit (GoF) measures (see Section 3.4). We use the SARIMA model as a benchmark because it is widely used in practice and all common statistics software like MATLAB®, R, or Python offer packages for an immediate application. The seasonal naive forecast often provides quite good estimates and is easy to implement. Hence, we use both as references for our model. Our new model is a deep-learning approach based on a convLSTM network, which requires a computationally expensive training step. In order to verify if this substantial complexity pays off regarding forecasting accuracy, we test the results also against two shallow approaches, namely a univariate single-layer LSTM model and a multivariate single-layer LSTM model.

As a basis we use climate data from various locations in Germany (see Section 3.1). Section 3.2 only concerns with the detailed setup of the applied neural networks, whereas Section 3.3 explains how we calibrate the SARIMA model and train the neural networks. Results are discussed in Section 4. All numerical computation is done with the software Python (3.6). The final model is set up with Tensorflow (API r1.13), loading and managing the data is done with the Python libraries pandas (v. 0.23.4) and numpy (v. 1.15.4). Visualization is done with matplotlib (v. 3.0.1). Data scaling and splitting is done with the scikit-learn library (v0.21.3). The complete source code is publicly available under the Github link https://github.com/michaelmunz/temperatureForecasting.

3.1. The data sets

We test multivariate data sets from five weather stations in Germany, namely Bremerhaven, Essen, Kassel, Ulm, and Kempten. All measurement stations face different conditions: The Bremerhaven station lies in a rather windy climate influenced by the sea — in contrary to the station in Kempten, which is situated near the Alps, the largest (and highest) mountain range in Europe. Ulm is a bit north of Kempten; hence, its climate is also partly influenced by the mountains but also by the river Danube, on whose banks it is situated. Essen, again, is part of the Ruhr area, a heavily populated industrial area in Western Germany.

Table 1
Summary of the sensor data used in the network input.

Factor	Unit	Factor	Unit
Air temperature	°C	Wind speed (at station altitude)	m/sec
Relative humidity	%	Relative air pressure (Δ_p)	mbar
Cloud coverage	0–8	Wind direction (at station altitude)	°
Hourly precipitation	mm		

Table 2
Summary of the input data.

Location	Station altitude (m)	Start	End	# Hours
Bremerhaven	1	2009-01–01	2013-12–31	42 719
Essen	150	2014-01–01	2018-12–31	43 495
Kassel	231	2009-01–01	2013-10–30	41 770
Ulm	567	2014-09–01	2018-12–31	36 547
Kempten	705	2014-01–01	2018-12–31	43 328

Kassel, a central German city, has no proximity to mountains or the sea. Data is collected by the German Climate Service DWD, which offers a wide range of freely available climate data via their ftp server (see www.dwd.de).

From the data pool of each weather station, we construct a seven dimensional data set. For a summary including units see Table 1. Thereby the relative air pressure Δ_p denotes the difference between the sensed air pressure at the station $p_{station}$ and the air pressure at sea level p_{sea_level}: $\Delta_p = p_{station} - p_{sea\_level}$. In Fig. 2 we exemplarily plot temperature, humidity, pressure, and wind speed of Ulm. Both temperature and pressure differences show a distinct seasonal pattern, whereby pressure differences are highest in winter. Relative humidity and wind speed, again, are rather noisy data and show no distinct seasonality.

A first analysis of the data indicates missing observations at various points of time, whereby the cloud coverage and the precipitation data show the largest deficits. Apart from that, if one input data set shows a missing value, the others are likely to do as well. For a consistent analysis, we remove the whole day if one or more hours contain some missing data points. Outliers are not corrected as we want to test also the forecasting methods robustness. Relevant features of the data sets are summed up in Table 2. Note that, due to different levels of availability, the time series length varies from data set to data set.

3.2. Neural network architectures

We test three different architectures of neural networks, one deep and two shallow ones with increasing complexity, which are described below. All chosen network architectures apply an end-to-end learning approach, because, while omitting the handcrafted feature extraction step, we can directly learn features from the data. This provides a powerful possibility for the model to adapt to different data. For example, the presence or absence of different sensors in a certain weather station does not lead to changing feature extraction steps. Furthermore, adaption of the feature extraction to different climate conditions or local weather situations is directly inferred from data. The recurrent connections of the LSTM networks, again, incorporate the time dependencies into the forecasting problem.

3.2.1. Univariate LSTM networks

A single-layer LSTM network with only the temperature as a single input sequence is used as a baseline reference. In doing so, we can directly compare its results to the SARIMA model, which is restricted to only one single input channel. The hyper parameters of the univariate LSTM network are shown in Table A.4.

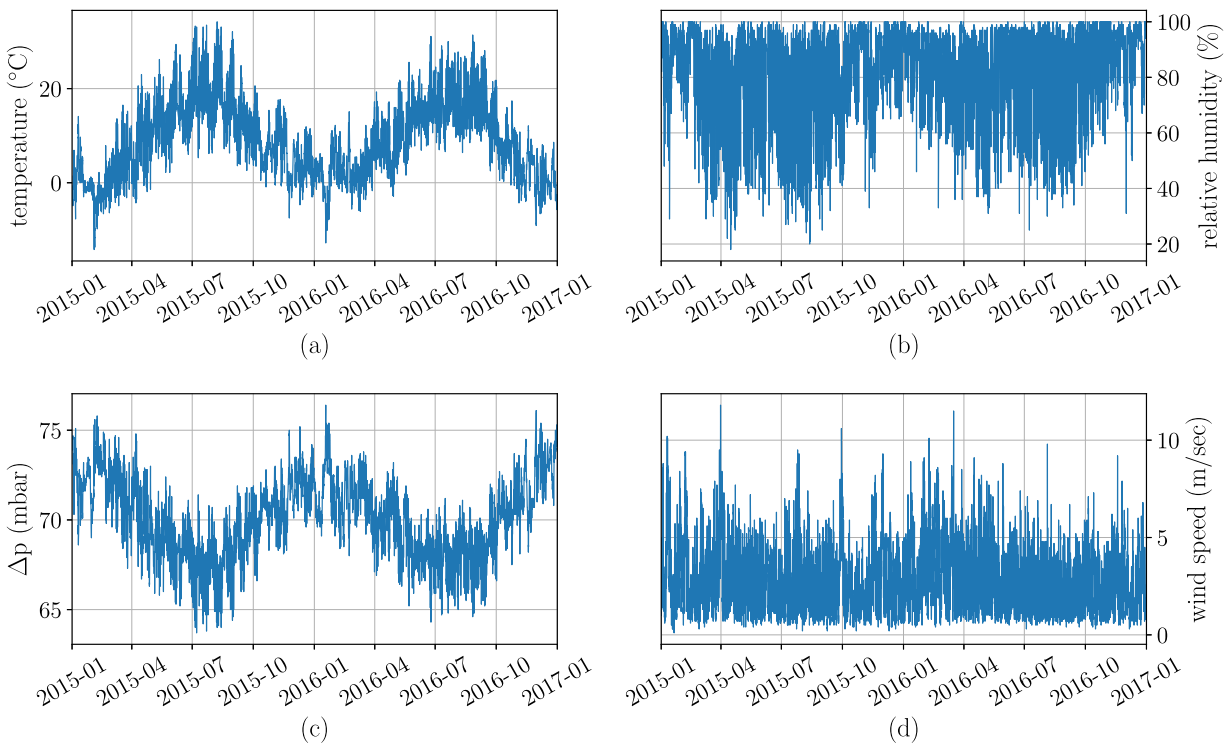


Fig. 2. Exemplary plots of Ulm's temperature (a), relative humidity (b), pressure difference (c) and wind speed (d) from 1st January 2015 to 1st January 2017, showing the interdependencies of the channels.

3.2.2. Multivariate LSTM networks

As each weather station provides multiple sensor data, which all might be helpful in forecasting temperature, we train a single layer LSTM network based on multivariate input data incorporating all available sensor data. This allows us to analyze the increase of forecast precision gained by incorporating multivariate data. Thereby, we use the same input data as for the convLSTM model (see below).

3.2.3. The convolutional LSTM network

The motivation for using a convLSTM network is (a) the assumption that individual measurements are highly correlated, hence the use of 2D-convolutional operations for feature extraction is valuable, and (b) the invariance of applying the convolutional kernels in time. In Fig. 6 the final setup of the convLSTM network is shown. The input to the convLSTM model is arranged in matrix form as described in Section 2.3.1 and fed to six convolutional layers, where the features are extracted. All features are then processed in two LSTM layers and two dense layers before being sent to the output layer. The output layer consists of 24 neurons, which represent the predicted temperature values in the chosen forecasting horizon. We are using L2-regularization in all layers and dropout only in the dense layers. Maxpooling is used after the second, fourth, and sixth convolutional layer. For a detailed overview of all network hyper parameters, please see Tables B.5 and B.6.

3.3. SARIMA model calibration and network training

For specifying the SARIMA model from Eq. (6) we initially compute the autocorrelation function (ACF) and the partial autocorrelation function (PACF) for all station. Thereby, the ACF is defined as $Corr(X_t, X_{t-k})$, and the PACF is defined as $Corr(X_t, X_{t-k}|X_{t-1}, \dots, X_{t-k+1})$ for $k \in \mathbb{N}$ (Box et al., 2016). Note that the ACF includes also indirect interdependencies. Consider an AR(1) process, i.e. a situation where there is only correlation between two subsequent observations X_t and X_{t-1}. As X_{t-1} is per definition correlated with X_{t-2} there is also some indirect correlation between X_t and X_{t-2} and the ACF value will be positive. Such indirect effects appear between any two points of time — also for autoregressive processes of higher order. Conclusions purely drawn from the ACF might not be unambiguous. Hence, we additionally exploit the PACF, which solely computes the correlation between X_t and X_{t-k}. Here, we explain the parameter choice for Ulm, results for all other stations are the same. With reference to Figs. 3 and 4 we see that differencing once is required and sufficient to eliminate all but the seasonal non-stationarities: All correlation values between ΔX_t and previous lags seem to be not significant — except for 24, which equals a seasonal lag of one day. This allows to set the differencing parameter d first mentioned in Eq. (5) to one and the seasonal parameter s from Eq. (6) to 24. From Fig. 4 we can also derive that the p from Eq. (1) is zero as there seems to be no correlation between two subsequent differences of observations. Mind that, as we differentiate once, we consider not the (P)ACF of X_t but of $\Delta X_t = X_t - X_{t-1}$. Parameter q from Eq. (2) is set to one to account for autocorrelation in the error term ϵ_t. Eventually, we yield non-biased parameter estimates, as the PACF after applying the final model to the data set shows no autocorrelation of significant size anymore (see Fig. 5). Given our analysis, the SARIMA$(0,1,1)\times(0,1,1)_{24}$ is the best choice. All relevant temporal dependencies have been effectively removed by the chosen parametrization. To produce the individual forecasts, we estimate the model parameters from a rolling time window of the past four weeks using (conditional) least squares estimation.

To compute the seasonal naive forecast, we merely identify the value of the same hour on the previous day, i.e. the forecast is computed as $\hat{X}_t = X_{t-24}$.

In case of the convLSTM network, to ensure a better generalization, the data is split day by day and shuffled, meaning that the order of the days, which are fed to the network, varies in every epoch. This method guarantees that the optimizer, which often reduces the learning rate in the course of training, does not get samples of a specific season of the year first. To compensate the risk of losing the annual seasonality and to simplify training, two additional channels are added to the networks input representation, specifically month and hour of day, according to every data point. Those additional channels represent domain knowledge about the problem. All data is scaled to [0,1].

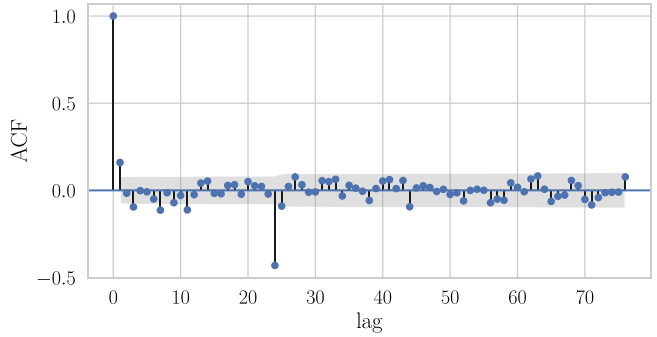


Fig. 3. Autocorrelation function of the first differences of Ulm's temperature data. The gray area marks the area where correlation values are not statistically significant.

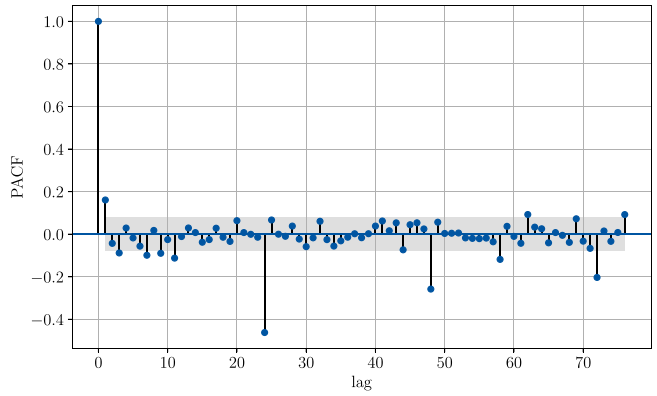


Fig. 4. Partial autocorrelation function of the first differences of Ulm's temperature data. The gray area marks the area where correlation values are not statistically significant.

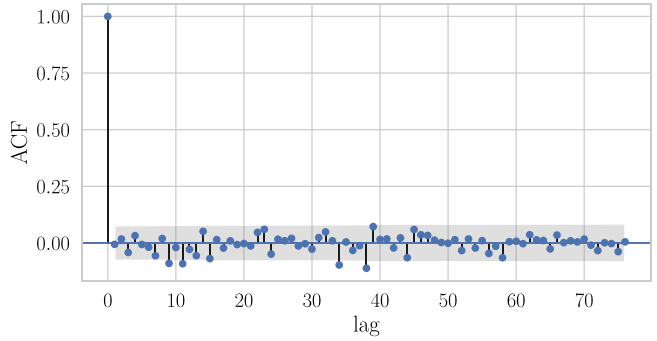


Fig. 5. Partial autocorrelation of the final model. The gray area marks the area where correlation values are not statistically significant.

For training of the network, the sliding window approach is used. Any window is represented as a matrix, where the rows correspond to the temporal course and the columns contain the different data channels. The window dimensions are 6×17, meaning that one column contains the values of the last six hours and 17 different data input channels are used. As the channel order is influencing the result, several test have been conducted. Finally, the following order is used in this work: air temperature, relative humidity, air pressure, cloud coverage, wind speed, wind direction, hourly precipitation, month, hour. The wind direction is partitioned into eight different sectors and one-hot encoded. After prediction, the window is shifted one hour, leading to an overlap of five hours. For calibrating the network, we apply the hold-out method: data is split into a training and test set, where 70% of the data is used as training set and the rest as test set. The training set itself is again split up into the actual training set (90% of it) and a validation set (the remaining 10%). We train 1000 epochs and use the model parameter set which reaches the best results based on the validation set. In case of the univariate LSTM model, only the temperature channel is fed into the network. The multivariate LSTM gets the same input data as the convLSTM, but the matrix is transformed into a sequence of vectors, as required by the LSTM network structure. The rest of the training is done accordingly to the process described above.

3.4. Goodness of fit measures

In order to compare the performance of our forecasting methods we compute two different GoF measures. The first one is the RMSE, which is defined as

$$RMSE = \sqrt{\frac{1}{T}\sum_{t=1}^{T}(\hat{X}_t - X_t)^2}. \tag{12}$$

given a forecast $\hat{X}_t$ for a temperature X_t. Besides, we compute the mean absolute scaled error (MASE), which compares a method's average absolute value to the average absolute value of the seasonal naive forecast. Its definition reads as follows:

$$MASE = \frac{\sum_{t=1}^{T}|\hat{X}_t - X_t|}{\sum_{t=1}^{T}|X_{t-s} - X_t|}, \tag{13}$$

for a seasonal lag s. The MASE is a simple and very comprehensible method, which allows to verify if it pays off to use a more complex method than the very simple seasonal naive forecast: A MASE value smaller than one means that the specific method is better than the naive forecast and vice versa. In this regard it is very similar to other forecasting skill evaluation metrics used in meteorology.

4. Results and discussion

4.1. Overall results

The major conclusions can be drawn purely from a visual comparison. In Fig. 7 we display forecast standard errors up to 24 h ahead. We see that the SARIMA model performs best in the first few hours followed by the univariate LSTM network. However, after a good start, the performance of both methods is significantly decreasing regarding RSME values, and they are clearly outperformed by both multivariate methods. Using more information, i.e. additional channels clearly pays off! Thereby we see that our convLSTM model performs best for all locations and most time horizons. The seasonal naive forecast, again, is the worst method for most prediction horizons. Only for 20 h–24 h forecasts its RMSE is at a similar level to the other method. This means, that for such horizons, the additional effort of the more complex alternative does not pay off. In Kempten, Kassel and Ulm, the univariate LSTM network shows a rather low performance over all time horizons, where it is even outperformed by the seasonal naive approach in the mid-range forecasting horizons.

To aggregate the graphical findings, we compute both MASE and RMSE for hourly temperature forecasts up to 24 h in advance and then calculate the mean over all values. The numerical results based on Ulm data are given in Table 3, whereby the best results are highlighted in bold letters. We see that, on average over all forecasting horizons, the convLSTM model performs best regarding both MASE and RMSE.

For detailed insight into the data, we plot temperature forecasts for Ulm on two sample days in Figs. 8 and 9. Thereby, for the sake of understanding, we split our set of forecasting methods into two parts. In Fig. 8 one day in August is displayed, showing some regular behavior. All methods capture the seasonal dynamics and generate good forecasts. In Fig. 9, again, one day in December is displayed, where we see a temperature drop. All methods include seasonal behavior and expect an increase of temperature in the morning, which was not the case. Hence, except for the multivariate LSTM model and the convLSTM model, the

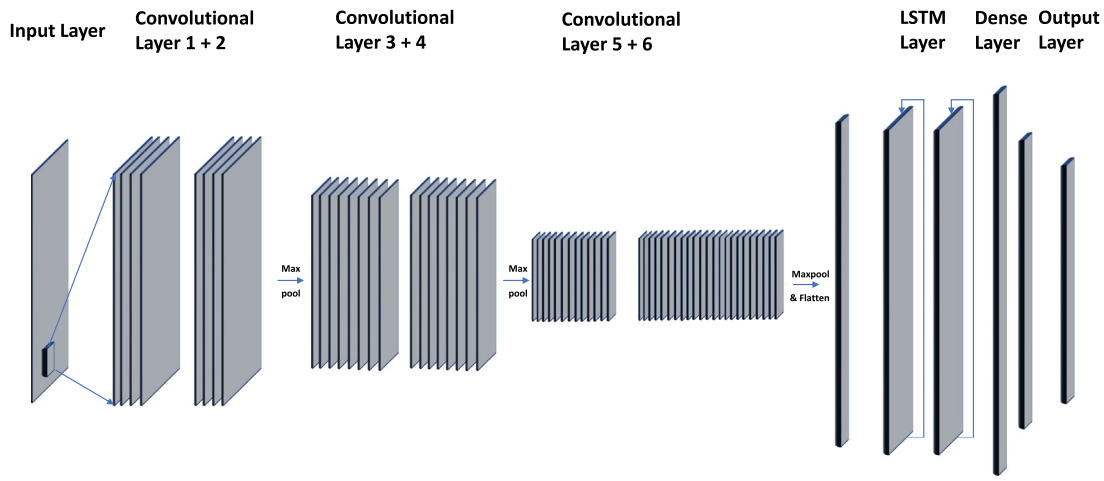
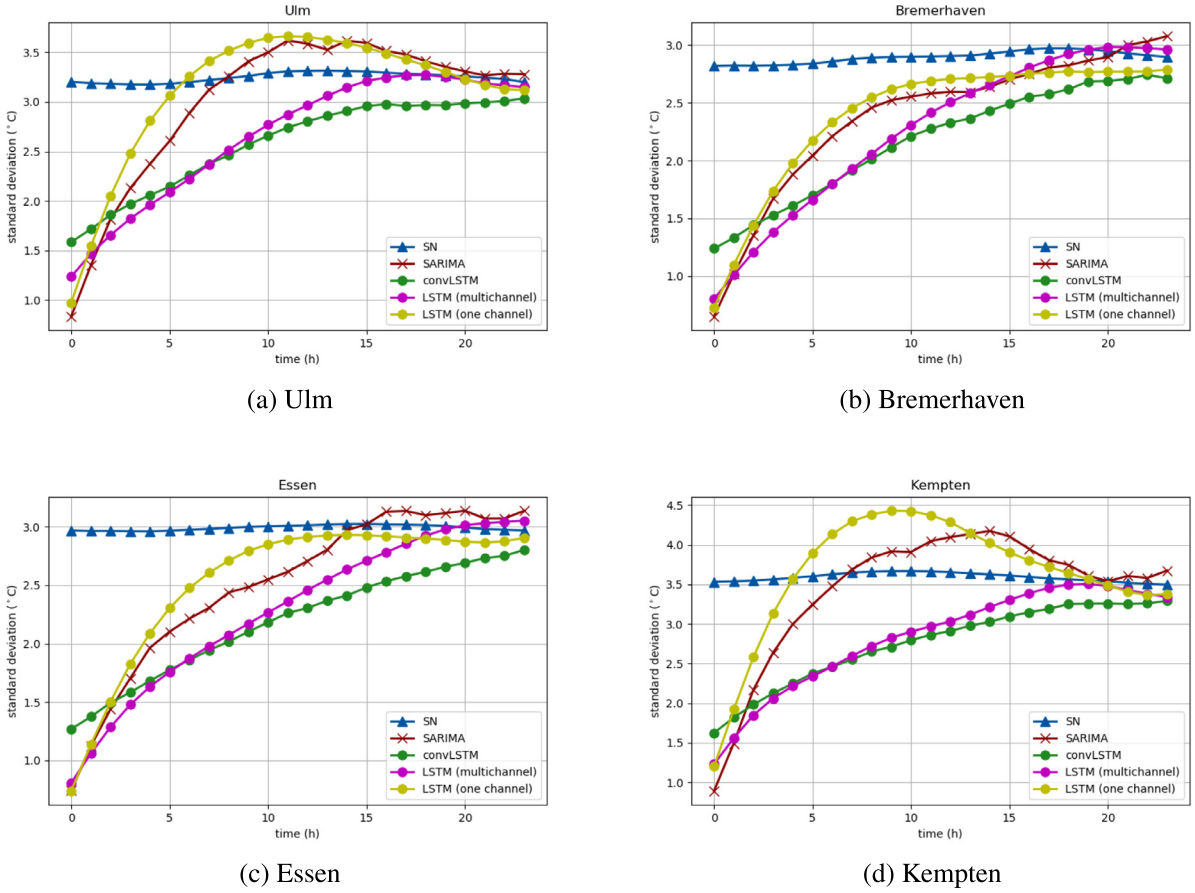


Fig. 6. Structure of the convolutional LSTM (convLSTM) network.



(a) Ulm (b) Bremerhaven

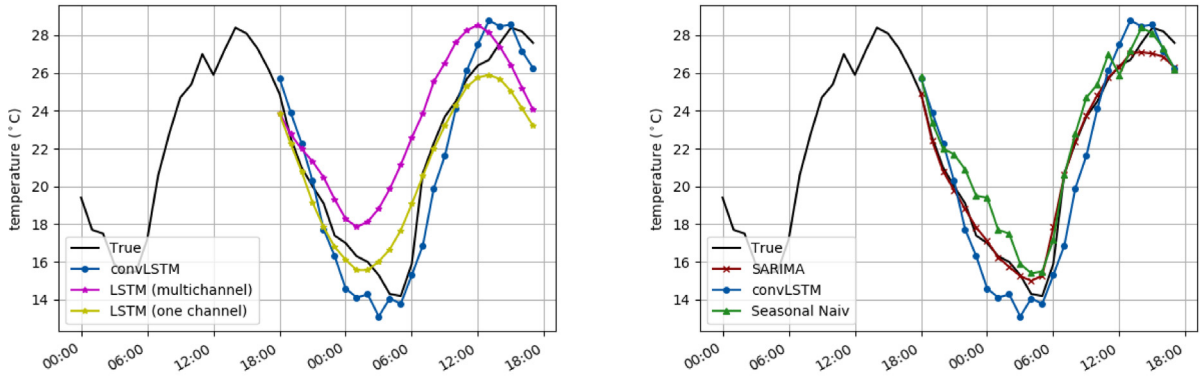
(c) Essen (d) Kempten

Fig. 7. Standard deviation, i.e. RMSE, for various forecast horizons of the seasonal naive forecast (SN), the SARIMA model, the multivariate and the univariate LSTM network, and the convLSTM network for different cities in Germany.

forecasts are in the wrong direction and need some time to correct. The SARIMA eventually incorporates the downwards trend, but both the seasonal naive approach and the (way more complex) univariate LSTM networks are completely off track. The convLSTM model clearly performs best, reacting to the structural changes but also expecting a seasonal development at the end. Hence, we have an increase in the forecasting error after about 6 a.m.

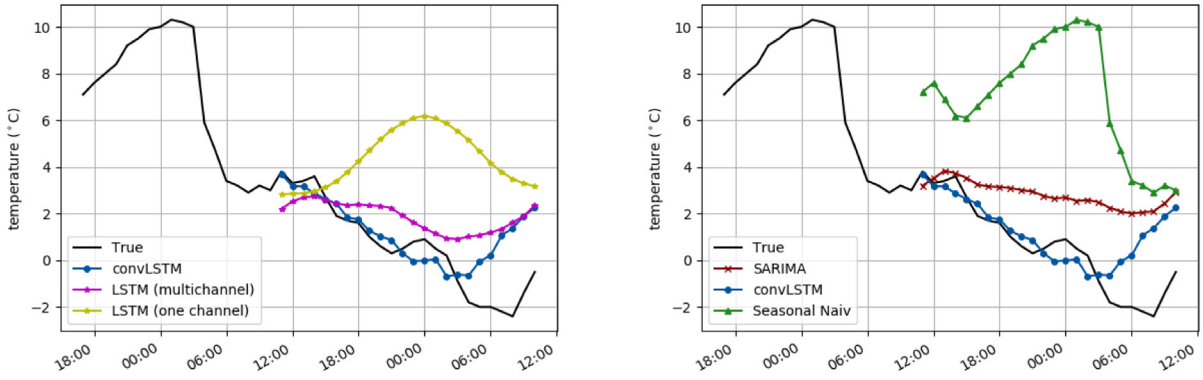
4.2. Discussion

Comparing univariate and multivariate LSTM networks, we see that using additional data channels is advantageous for temperature forecasting: Although the structure and training procedure has not been changed, the multivariate LSTM model performs considerably better regarding RMSE than the univariate LSTM model. The networks are



(a) Forecasting methods, part 1 (b) Forecasting methods, part 2

Fig. 8. Temperature forecasts for Ulm on 20th August 2018 generated by the seasonal naive forecast (SN), the SARIMA model, the multivariate and the univariate LSTM network, and the convLSTM network.



(a) Forecasting Methods, Part 1 (b) Forecasting Methods, Part 2

Fig. 9. Temperature forecasts for Ulm on 24th December 2018 generated by the seasonal naive forecast (SN), the SARIMA model, the multivariate and the univariate LSTM network, and the convLSTM network.

Table 3
Average RMSE and MASE for hourly temperature forecasts up to 24 h in advance based on data from Ulm. Considered models were the naive approach, SARIMA, univariate LSTM (LSTM), multivariate LSTM (LSTM(multi)) and convLSTM.

Model	Naive	SARIMA	LSTM	LSTM(multi)	convLSTM
MASE	1.00	1.05	1.18	0.99	**0.93**
RMSE	2.87	2.55	2.83	2.37	**2.10**

obviously capable of handling the dependency of future temperature on parameters such as air pressure or humidity. The same holds true for the SARIMA model, which is only capable of using one data input channel. When comparing the multivariate LSTM network with the convolutional LSTM model we see that using a deep network leads to even more precise results. The convLSTM model performs best after approximately six hours. For shorter forecast horizons, it might be better to use a simpler model, such as the multivariate LSTM model. But one very important point to notice is that having a closer look at temperature patterns which do not follow the seasonal trend (as exemplarily shown in Fig. 9), the following conclusion can be drawn: Given the results from Fig. 7 one might think that the seasonal naive approach is a fairly good alternative for 20 h–24 h forecasts. However, this holds true only for two succeeding days with similar weather patterns. If the weather is changing – and this is the situation where good temperature forecasts are needed most – the model's accuracy is quite weak. As all the models have been applied to data from different cities over Germany, one can conclude that the methods are applicable to different conditions and locations. This again shows that the chosen end-to-end training methodology is beneficial for the described forecasting scenario. However, despite we have shown that the convLSTM network performs best given our GoF measures, there is the chance to improve on these results. As discussed above there are times (constant weather without e.g. temperature drops) or forecasting horizons (less than six hours) where other models like the SARIMA show a reasonable performance. Therefore, a fusion of models in the context of an ensemble forecast could improve the performance of the forecast. A recently published study Abdel-Nasser et al. (2020) show the applicability of the Choquet integral for aggregating different LSTM networks. This approach is not in the scope of this study, which focuses on presenting an application for the convLSTM model, but might be applicable to the proposed convLSTM network in combination with SARIMA or other convLSTM models.

5. Conclusion

In this paper, we propose a new method for temperature forecasting, which is based on a convolutional LSTM neural network. As benchmarks we apply the classic SARIMA model, the seasonal naive approach, and two simpler neural networks. For comparing all methods, we empirically apply them to temperature data from five cities in

Table A.4
Hyper parameters of the LSTM network.

Category	Parameter	Value	Description
Data	Test set size	30%	Percentage of data used for testing
Data	Validation set size	10%	Percentage of data used for validation
Data	Windowlength	1 h	Length of one window
Data	Windowshift	1	Amount of data points one window is shifted
Data	Sequencelength	24	Amount of windows being held together (shuffling the whole package)
Data	Prediction length	24	Forecasted hours
Data	Inputs	1 or 17	Temperature, relative humidity, ΔP, p0, cloudiness, wind speed, wind direction (one-hot encoded), precipitation, month, hour of day
network- LSTM	Neurons	32	Amount of neurons
network- LSTM	Layer	1	Amount of layers
network- Dense2	Neurons	24	Amount of neurons (output)
Network	Learning rate	Keras default	Learning rate
Training	Loss	MSE	Mean squared error (+regularization term)
Training	Optimizer	AdamOptimizer	Method to optimize
Training	Initializer	Keras default	Method to initialize weights
Training	Epochs	200	

Table B.5
Hyper parameters of the convolutional LSTM network (1).

Category	Parameter	Value	Description
network - conv1	Filter	8	Amount of filters
network - conv1	Kernel size	3 × 3	Size of a filter
network - conv1	Stride	1	Filter shift
network - conv1	Padding	Same	Treatment of image borders
network - conv2	Filter	8	Amount of filters
network - conv2	Kernel size	2 × 2	Size of a filter
network - conv2	Stride	1	Filter shift
network - conv2	Padding	Same	Treatment of image borders
network - maxpool	Kernel size	2 × 2	Size of a maxpool-kernel
network - conv3	Filter	16	Amount of filters
network - conv3	kernel size	2 × 2	Size of a filter
network - conv3	Stride	1	Filter shift
network - conv3	Padding	Same	Treatment of image borders
network - conv4	Filter	16	Amount of filters
network - conv4	Kernel size	3 × 3	Size of a filter
network - conv4	Stride	1	Filter shift
network - conv4	Padding	Same	Treatment of image borders
network - maxpool	Kernel size	2 × 2	Size of a maxpool-kernel
network - conv5	Filter	32	Amount of filters
network - conv5	Kernel size	3 × 3	Size of a filter
network - conv5	Stride	1	Filter shift
network - conv5	Padding	Same	Treatment of image borders
network - conv6	Filter	64	Amount of filters
network - conv6	Kernel size	2 × 2	Size of a filter
network - conv6	Stride	1	Filter shift
network - conv6	Padding	Same	Treatment of image borders
network- LSTM	Neurons	24	Amount of neurons
network- LSTM	Layer	2	Amount of layers
network- Dense1	Neurons	512	Amount of neurons
network- Dense2	Neurons	64	Amount of neurons
network- Dense2	Neurons	24	Amount of neurons (output)
Network	Dropout	0.2	Dropout rate (in the dense layer and in the LSTM)
Network	l2 regularization	0.0001	L2 reg parameter
Network	Learning rate	0.00005	Learning rate
Training	Loss	MSE	Mean squared error (+ regularization term)
Training	Optimizer	AdamOptimizer	Method to optimize
Training	Initializer	xavier_initializer	Method to initialize weights
Training	Epochs	3000	Number of training epochs

Table B.6
Hyper parameters of the convolutional LSTM network (2).

Category	Parameter	Value	Description
Data	Test set size	30%	Percentage of data used for testing
Data	Validation set size	10%	percentage of data used for validation
Data	Windowlength	6 h	Length of one window
Data	Windowshift	1	Amount of data points one window is shifted
Data	Sequencelength	24	Amount of windows being held together (shuffling the whole package)
Data	Prediction length	24	Forecasted hours
Data	Inputs	17	Temperature, relative humidity, Δp, p0, cloudiness, wind speed, windDirection (one-hot encoded), precipitation, month, hour of day
Data	Batchsize	24	24 samples per batch

Germany. Even though, altogether the deep convLSTM network shows better results, feeding more information is not always beneficial — we see that it is often outperformed by simpler models in the first few hours on average. However, for larger forecasting horizons, the deep network delivers more accurate results. Especially in weather situation, where temperature changes rapidly, the convLSTM network outperforms other methods by well adapting to changed input. Regarding prediction time, neural networks are independent of the latest data and have to be trained only once. Once the training is done, it only requires data from the last six hours for prediction, which is the size of the sliding window. To sum up our findings: The convLSTM network shows promising results especially for forecasting horizons between 6 h and 24 h and in case of unexpected weather changes as it utilizes extra information compared to univariate models. For shorter forecasting horizons, simpler models might also be useful. Overall, as weather data can easily be captured and deep neural networks can be efficiently trained using GPGPUs, the applicability of convLSTM networks is beneficial for temperature forecasting. Nevertheless, this study has its limits, and future research is required in various direction. First, one has to verify if changing the time granularity would produce better results. Second, the relevance of the various input channels has to be analyzed to prevent negative influence on the outcome of an individual channel. Moreover, involving additional measurements, which are more sensitive for the prediction of upcoming temperature changes, might reduce the effect of time delay in prediction of temperature drops in the network. Besides, although the data originates from a number of different places, it is limited to Germany, hence to a moderate continental European climate. Future work needs to verify our results using global weather data. Since our model is especially capable of handling sudden weather changes, we expect a fairly good performance in locations suffering from extreme weather conditions. For a more general benchmark the model selection can also be extended, especially more complex ensemble forecasts should be tested. Another issue that will be evaluated in the future is the robustness of the model when dealing with incomplete data.

CRediT authorship contribution statement

David Kreuzer: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Data curation, Writing - review & editing, Visualization. **Michael Munz:** Conceptualization, Methodology, Validation, Resources, Writing - review & editing, Supervision. **Stephan Schlüter:** Conceptualization, Methodology, Validation, Resources, Data curation, Writing - original draft, Writing - review & editing, Supervision, Project administration.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

We had like to thank the Deutscher Wetterdienst for the access to their climate data center. Without this huge data basis, our research would not have been possible.

Appendix A. Hyper parameters of the LSTM network

See Table A.4.

Appendix B. Hyper parameters of the convolutional LSTM network

See Tables B.5 and B.6.

References

Abdel-Nasser, M., & Mahmoud, K. (2019). Accurate photovoltaic power forecasting models using deep LSTM-RNN. *Neural Computing and Applications*, *31*(7), 2727–2740. http://dx.doi.org/10.1007/s00521-017-3225-z.

Abdel-Nasser, M., Mahmoud, K., & Lehtonen, M. (2020). Reliable solar irradiance forecasting approach based on choquet integral and deep LSTMs. *IEEE Transactions on Industrial Informatics*.

Alhirmizy, S., & Qader, B. (2019). Multivariate time series forecasting with LSTM for madrid, Spain pollution. In *The IEEE international conference on computing, information science technology and their applications - 2019* (pp. 1–5). [Piscataway, New Jersey]: IEEE, http://dx.doi.org/10.1109/ICCISTA.2019.8830667.

Box, G. E. P., Jenkins, G. M., Reinsel, G. C., & Ljung, G. M. (2016). Time series analysis: Forecasting and control, In *Wiley Series in Probability and Statistics*, (Fifth edition). Hoboken, New Jersey: John Wiley & Sons Inc.

Bundesnetzagentur (2017). Netz- und Systemsicherheitsmaß nahmen gesamtjahr und viertes quartal. URL: https://www.bundesnetzagentur.de/.

Curceac, S., Ternynck, C., Ouarda, T. B., Chebana, F., & Niang, S. D. (2019). Short-term air temperature forecasting using nonparametric functional data analysis and SARMA models. *Environmental Modelling & Software*, *111*, 394–408. http://dx.doi.org/10.1016/j.envsoft.2018.09.017.

Dong, D., Sheng, Z., & Yang, T. (2018). Wind power prediction based on recurrent neural network with long short-term memory units. (pp. 34–38). http://dx.doi.org/10.1109/REPE.2018.8657666.

Frnda, J., Durica, M., Nedoma, J., Zabka, S., Martinek, R., & Kostelansky, M. (2019). A weather forecast model accuracy analysis and ECMWF enhancement proposal by neural network. *Sensors (Basel, Switzerland)*, *19*(23), http://dx.doi.org/10.3390/s19235144.

Graf, R., Zhu, S., & Sivakumar, B. (2019). Forecasting river water temperature time series using a wavelet–neural network hybrid modelling approach. *Journal of Hydrology*, *578*, Article 124115. http://dx.doi.org/10.1016/j.jhydrol.2019.124115.

Green, M. (2003). General temperature dependence of solar cell performance and implications for device modeling. *Progress in Photovoltaics: Research and Applications*, *11*, 333–340. http://dx.doi.org/10.1002/pip.496.

Hassani, H., Silva, E. S., Gupta, R., & Das, S. (2018). Predicting global temperature anomaly: A definitive investigation using an ensemble of twelve competing forecasting models. *Physica A. Statistical Mechanics and its Applications*, *509*, 121–139. http://dx.doi.org/10.1016/j.physa.2018.05.147.

Herwartz, H., & Schlüter, S. (2016). On the predictive information of futures' prices: A wavelet-based assessment. *Journal of Forecasting*, http://dx.doi.org/10.1002/for.2435.

Heryadi, Y., & Warnars, H. L. H. S. (2017). Learning temporal representation of transaction amount for fraudulent transaction recognition using CNN, stacked LSTM, and CNN-LSTM. In *The 2017 international conference on cybernetics and computational intelligence* (pp. 84–89). Piscataway, NJ: IEEE, http://dx.doi.org/10.1109/CYBERNETICSCOM.2017.8311689.

Hochreiter, S. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780. http://dx.doi.org/10.1162/neco.1997.9.8.1735.

Hochreiter, S. (1998). The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, *06*(02), 107–116. http://dx.doi.org/10.1142/S0218488598000094.

Kann, A., Haiden, T., & Wittmann, C. (2011). Combining 2-m temperature nowcasting and short range ensemble forecasting. *Nonlinear Processes in Geophysics*, *18*(6), 903–910. http://dx.doi.org/10.5194/npg-18-903-2011.

Kim, Y. (2014). Convolutional neural networks for sentence classification. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)* (pp. 1746–1751). Stroudsburg, PA, USA: Association for Computational Linguistics, http://dx.doi.org/10.3115/v1/D14-1181.

Kim, T.-Y., & Cho, S.-B. (2019). Predicting residential energy consumption using CNN-LSTM neural networks. *Energy*, *182*, 72–81. http://dx.doi.org/10.1016/j.energy.2019.05.230.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, *60*(6), 84–90. http://dx.doi.org/10.1145/3065386.

Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324. http://dx.doi.org/10.1109/5.726791.

Lee, S., Lee, Y.-S., & Son, Y. (2020). Forecasting daily temperatures with different time interval data using deep neural networks. *Applied Sciences*, *10*(5), 1609. http://dx.doi.org/10.3390/app10051609.

Liu, D., Xie, S., Li, Y., Zhao, D., & El-Alfy, E.-S. M. (2017). *Lecture Notes in Computer Science*: *vol. 10635*, *Neural information processing: 24th international conference, ICONIP 2017, Guangzhou, China, November 14-18, 2017, proceedings, Part II* (pp. 198–206). Cham: Springer International Publishing, http://dx.doi.org/10.1007/978-3-319-70096-0.

Lopez, E., Valle, C., & Allende, H. (2016). Recurrent networks for wind speed forecasting. In *International Conference on Pattern Recognition Systems (ICPRS-16)* (pp. 1–6).

Lorenz, E. (1963). Deterministic nonperiodic flow. *Journal of the Atmospheric Sciences*, *20*(2), 130–141. http://dx.doi.org/10.1175/1520-0469(1963)020<0130:DNF>2.0.CO;2.

McNeil, A. J., Frey, R., & Embrechts, P. (2015). Quantitative risk management: concepts, techniques and tools, In *Princeton series in finance*, (Revised edition). Princeton University Press.

Molano-Mazon, M., Onken, A., Piasini, E., & Panzeri, S. (2018). Synthesizing realistic neural population activity patterns using generative adversarial networks.

Möller, A., & Groß, J. (2016). Probabilistic temperature forecasting based on an ensemble autoregressive modification. *Quarterly Journal of the Royal Meteorological Society*, *142*(696), 1385–1394. http://dx.doi.org/10.1002/qj.2741.

Ordóñez, F., & Roggen, D. (2016). Deep convolutional and LSTM recurrent neural networks for multimodal wearable activity recognition. *Sensors (Basel, Switzerland)*, *16*(1), http://dx.doi.org/10.3390/s16010115.

Qin, D., Yu, J., Zou, G., Yong, R., Zhao, Q., & Zhang, B. (2019). A novel combined prediction scheme based on CNN and LSTM for urban pm 2.5 concentration. *IEEE Access*, *7*, 20050–20059. http://dx.doi.org/10.1109/ACCESS.2019.2897028.

Schlüter, S., & Deuschle, C. (2014). Wavelet-based forecasting of ARIMA time series - an empirical comparison of different methods. *Managerial Economics*, *15*(1), 107. http://dx.doi.org/10.7494/manage.2014.15.1.107.

Shi, X., Chen, Z., Wang, H., Yeung, D.-Y., Wong, W.-k., & Woo, W.-c. (2015). Convolutional LSTM network: A machine learning approach for precipitation nowcasting.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2015). Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 1–9). Piscataway, NJ: IEEE, http://dx.doi.org/10.1109/CVPR.2015.7298594.

Tran, D., Bourdev, L., Fergus, R., Torresani, L., & Paluri, M. (2015). Learning spatiotemporal features with 3D convolutional networks. In *2015 IEEE International Conference on Computer Vision* (pp. 4489–4497). Piscataway, NJ: IEEE, http://dx.doi.org/10.1109/ICCV.2015.510.

Xiaoyun, Q., Xiaoning, K., Chao, Z., Shuai, J., & Xiuda, M. (2016). Short-term prediction of wind power based on deep long short-term memory. In *2016 IEEE PES Asia-Pacific Power and Energy Engineering Conference (APPEEC)* (pp. 1148–1152).

Xue, N., Triguero, I., Figueredo, G. P., & Landa-Silva, D. (2019). Evolving deep CNN-LSTMs for inventory time series prediction. In *2019 IEEE Congress on Evolutionary Computation (CEC)* (pp. 1517–1524). IEEE, http://dx.doi.org/10.1109/CEC.2019.8789957.

Zhang, Q., Wang, H., Dong, J., Zhong, G., & Sun, X. (2017). Prediction of sea surface temperature using long short-term memory. *IEEE Geoscience and Remote Sensing Letters*, *14*(10), 1745–1749. http://dx.doi.org/10.1109/LGRS.2017.2733548.

Zhou, K., Zheng, Y., Li, B., Dong, W., & Zhang, X. (2019). Forecasting different types of convective weather: A deep learning approach. *Journal of Meteorological Research*, *33*(5), 797–809. http://dx.doi.org/10.1007/s13351-019-8162-6.