From Precipitation to Prediction: Integrating ConvLSTM Models for Comprehensive River Runoff Forecasting

Florian Börgel[1], **Sven Karsten**[1], **Karoline Rummel**[1]

[1]Leibniz-Institute for Baltic Sea Research Warnemünde,

Corresponding author: Florian Börgel, `florian.boergel@io-warnemuende.de`

Abstract

a

Plain Language Summary

b

1 Introduction

River runoff is an important component of the global water cycle as it comprises about one third of the precipitation over land areas (Hagemann et al., 2020). In the context of climate change studies, river runoff is usually generated in two ways. First, river runoff as input for ocean models can be created using hydrological models such as the Hydrological Discharge (HD) model (Hagemann et al., 2020). HD models calculates the water balance using hydrological processes (e.g. snow, glaciers, soil moisture, groundwater contribution). It represents a complex forecasting tool that uses underlying physical processes. A different approach would use data-based models that integrate statistical correction, using the land surface schemes of global or regional climate models.

The relatively recent emergence of machine learning (ML) models has offered different approaches to river runoff forecasting. Accurate runoff forecasting is crucial for effective water resources management, particularly over extended periods (W. Fang et al., 2019; Tan et al., 2018). Common approaches employ feed-forward artificial neural networks, support vector machines, adaptive neuro-fuzzy inference systems, and notably, Long Short-Term Memory (LSTM) neural networks that have gained traction for long-term hydrological forecasting due to their excellent performance (Ashrafi et al., 2017; L. Fang & Shao, 2022; Huang et al., 2014; Humphrey et al., 2016; Kratzert et al., 2018; Liu et al., 2020).

LSTM networks, first introduced by Hochreiter & Schmidhuber (1997) , are an evolution of the classical Recurrent Neural Networks (RNNs). Their structure enables them to learn long-term dependencies while avoiding the vanishing or exploding gradient problem. They have shown stability and efficacy in sequence-to-sequence predictions. However, a limitation of LSTMs is their inability to effectively capture two-dimensional structures, an area where Convolutional Neural Networks (CNNs) excel. Recognizing this limitation SHI et al. (2015) proposed a Convolutional LSTM (ConvLSTM) architecture, which combines the strengths of both LSTM and CNN. The ConvLSTM network has been proven useful for spatio-temporal applications such as precipitation nowcasting (SHI et al., 2015), flood forecasting (Moishin et al., 2021), and river runoff forecasting (Ha et al., 2021; Zhu et al., 2023).

In this work, we demonstrate that ConvLSTM networks are a reliable method for predicting multiple rivers simultaneously, using only atmospheric forcing, even in the absence of a fully functional hydrological model with a complex parameterization. We use the Baltic Sea catchment as an example to illustrate our approach. Although the methodology we propose is universally applicable across various geographic regions, the Baltic Sea presents a challenge due to its unique hydrological characteristics, being nearly decoupled from the open ocean (see Figure). As a consequence, the salinity of the Baltic Sea is driven to a large part by freshwater supply from rivers. Freshwater enters the Baltic Sea through river runoff or positive net precipitation (precipitation minus evaporation) over the sea surface. The net precipitation accounts for 11 % and the river input for 89 % of the total freshwater input (Meier & Döscher, 2002). Modelling the Baltic Sea is therefore to a large part the result of the quality of the river input, that is used for the simulation. This makes the accurate modeling of river runoff especially critical for simulations pertaining to the Baltic Sea.

In this work we will, we present a ConvLSTM architecture that is able to predict daily river runoff for 97 rivers across the Baltic Sea catchment. ***mehr Fokus auf Neues?***

2 Implemented model architecture

2.1 The main idea

We assume that the runoff for a specific point in time for all N_r considered rivers, collected in the vector $\vec{R}^t \in \mathbb{R}^{N_r}$, can be accurately approximated by a functional $\vec{M}(\{X_k^t[x,y,\tau]\})$ of $k = 1,\dots N_k$ atmospheric fields $X_k^t[x,y,\tau]$ that are known for the past $\tau = 1,\dots N_\tau$ time instances, i.e.

$$\vec{R}^t = \vec{M}(\{X_k^t[x,y,\tau]\}) \ .$$

The fields are evaluated on a spatial domain $x = 1,\dots N_x$ and $y = 1,\dots N_y$ that is sufficiently large to capture all significant local and non-local contributions of atmospheric fields to the river runoff.

Usually such a mapping is realized in a suitable hydrological model that simulates all relevant physical processes that transform variables like, e.g., precipitation and evaproration to the river runoff, relying on expertise for tuning all appearing parameters to reasonable values.

As an alternative, we propose that this functional can be adequately represented by a combination of a ConvLSTM with a subsequent FC network. That way, there is no need to have knowledge of involved processes and how they can be modeled, instead, these features can be "lerned" by the network in an automated manner. Our poroposed network architecture is visualized in Figure 1 and described in detail in the following. In order to provide an overview, we will go through the main ingredients of this architecture one-by-one.

2.2 ConvLSTM network

2.2.1 The LSTM approach

Before turning directly to the convolutional LSTM, the simpler architecture of the plain LSTM is examined and should serve as a role model for the later consideration of the full ConvLSTM.

The Long Short-Term Memory (LSTM), a specialized form of Recurrent Neural Networks (RNNs), is specifically tailored for modeling temporal sequences $\vec{X}^t[1],\dots\vec{X}^t[\tau],\dots\vec{X}^t[N_\tau]$ of N_τ input quantities $\vec{X}^t[\tau] = (X_k^t[\tau]) \in \mathbb{R}^{N_k}$. The sequence is taken from a dataset given in form of a time series $\{\vec{X}^t\}$ and the endpoint should coincide with the specific element in the time series connected to time t, i.e. $\vec{X}^t[N_\tau] \equiv \vec{X}^t$, see Figure 1. The N_k is the number of used input "channels" and can, for example, represent different measurable quantities. Its unique design allows it to adeptly handle long-range dependencies, setting it apart from traditional RNNs in terms of accuracy (see Figure 2).

This performance in modeling long-range dependencies has been validated in various studies (**XXX?**). The key component of the LSTM's innovation is its cell state, $\vec{C}^t[\tau] = (C_h^t[\tau]) \in \mathbb{R}^{N_h}$, which stores state information, also referred to as *long-term memory*. This state information complements the so-called hidden state $\vec{H}^t[\tau] = (H_h^t[\tau]) \in \mathbb{R}^{N_h}$ vector, which is also known from simpler neural network architectures. In case of the LSTM, the hidden state vector plays the role of the *short-term meory*.

A significant advantage of this architecture is the memory cell's ability to retain gradients. This mechanism addresses the vanishing gradient problem, where, as input sequences elongate, the influence of initial stages becomes harder to capture, causing gradients of early input points to approach zero. The LSTM's activation function,

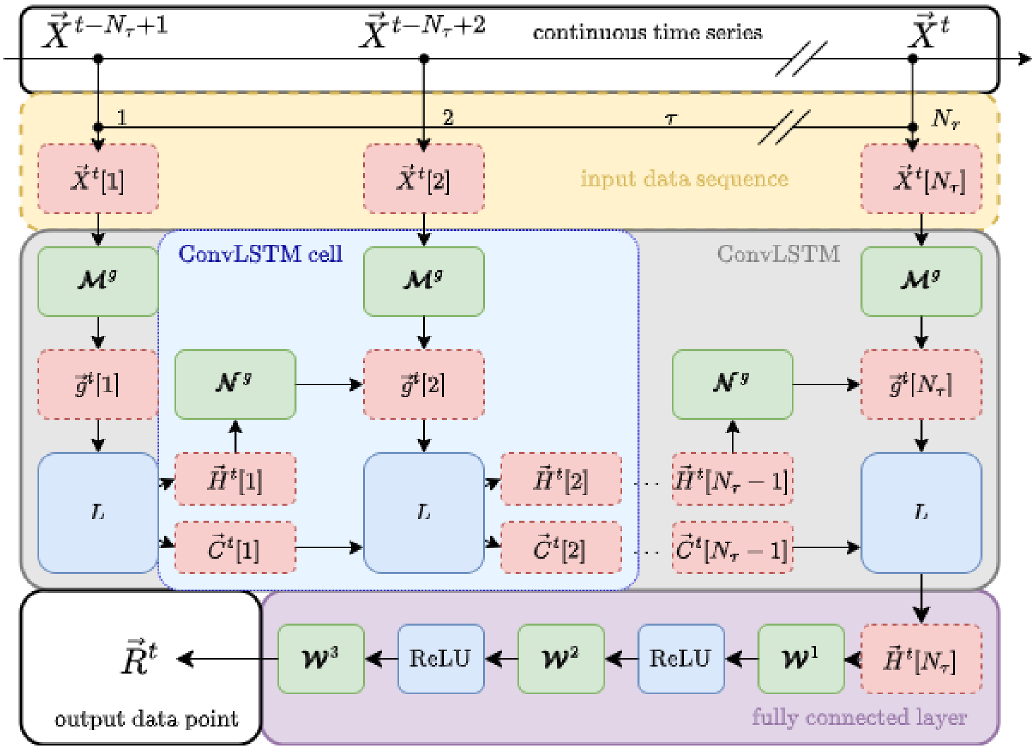


Figure 1: **Combined ConvLSTM and FC network architecture**. The starting point is the continuous timeseries of input data $\{\vec{X}^t\}$, upper white block. From this series, a contiguous sequence of N_τ elements (yellow block) is used to feed a chain of N_τ connected ConvLSTM cells (light blue block) building the ConvLSTM network (grey block). The input seuqence is mapped via weighting matrices $\mathcal{M}^g$ (green blocks) onto gate vectors $\vec{g}^t[\tau]$. The gate vectors are then used to update the cell state $\vec{C}^t[\tau-1]$ and the hidden state $\vec{H}^t[\tau-1]$ of the last ConvLSTM cell to the current values $\vec{C}^t[\tau]$ and $\vec{H}^t[\tau]$, respectively. The update is performed with LSTM core equations collectively described by the mapping L, see Equation 2. The weighting matrices $\mathcal{N}^g$ (green blocks) control how much of the last hidden state enters the updated state. The final output of the ConvLSTM $\vec{H}^t[\tau]$ is then propagated to a FC network, which itself is a chain of three FC layers consisiting of weigthing matrices $\mathcal{W}$ and connected via ReLU functions, see Section 2.3. The final result is then the river runoff $\vec{R}^t$ for all rivers considered at the current time instance t (white block on the lower left). Note that all bias vectors are ommitted for the sake of clarity. See text for more information.

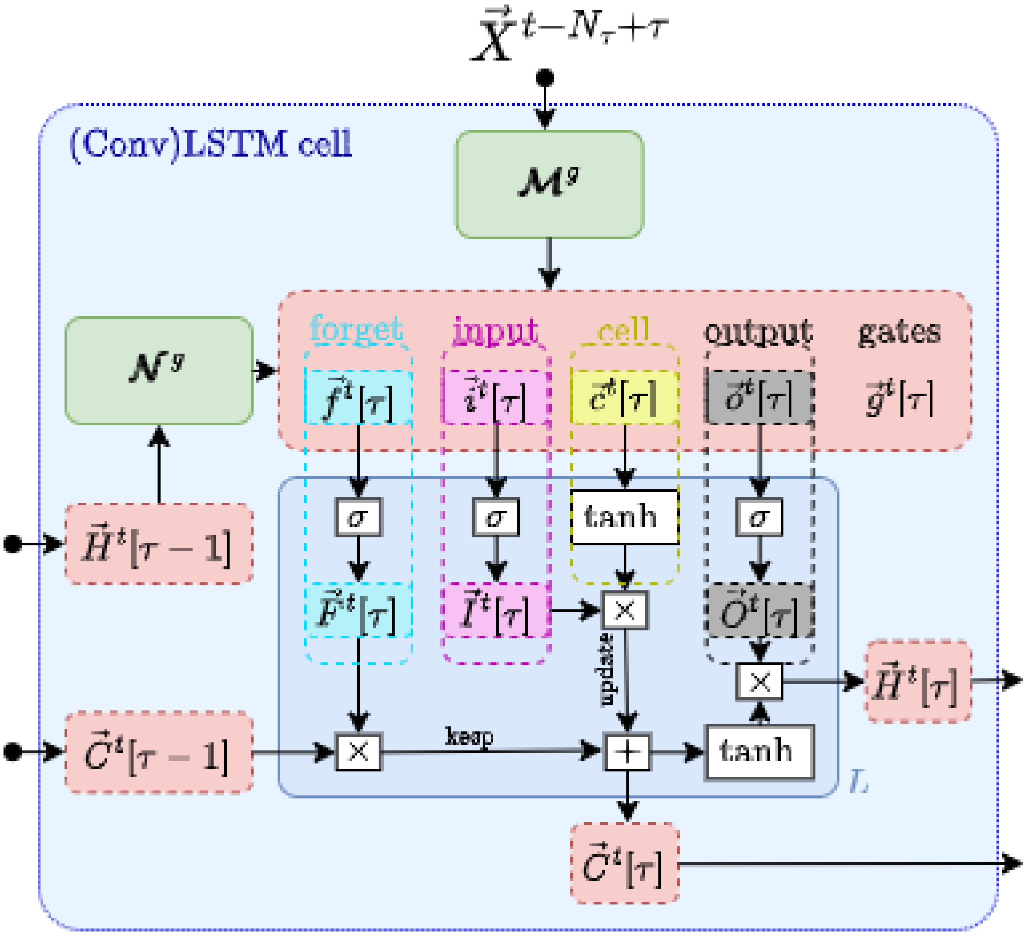


Figure 2: **Inner structure of a Long Short-Term Memory Cell**. See Figure 1 and text for information.

inherently recurrent, mirrors the identity function with a consistent derivative of 1.0, ensuring the gradient remains stable throughout backpropagation.

The cell state and the hidden state are vectors, where each element is associated with one of the N_h hidden layers, labeled by h, which are internal, *artificial* degrees of freedom that enable the high adaptibility of neural networks. These two state vectors are determined through several self-parameterized gates, all in the same vector space as $\vec{C}^t[\tau]$, see Figure 2 for a visualization.

In particular, the forget gate $\vec{F}^t[\tau]$ defines the portion of the previous (long-term memory) cell state $\vec{C}^t[\tau - 1]$ that should be kept, see dashed cyan box therin. The input gate $\vec{I}^t[\tau]$ controls the contribution of the current input used to update the long-term memory, $\vec{C}^t[\tau]$ (magenta and yellow boxes). The output gate, $\vec{O}^t[\tau]$, then determines how much of this updated long-term memory contributes to the new (short-term memory) hidden state, $\vec{H}^t[\tau]$ (black dashed box).

For a fixed point τ in the sequence, the action of a LSTM cell, i.e. the connection between the input $\vec{X}^t[\tau]$, the various gates and the state vectors, is mathematically given as follows.

First, the elements of the input sequence together with the hidden state are mapped onto auxiliary gate vectors, collectively denoted by $\vec{g}^t[\tau] = (g_h^t[\tau]) \in \mathbb{R}^{N_h}$, via

$$g_h^t[\tau] = \mathcal{M}_{hk}^g \, X_k^t[\tau] + \mathcal{N}_{hh'}^g \, H_{h'}^t[\tau - 1] + \mathcal{B}_h^g \ ,$$

where $g = i, f, o, c$ stands for the input, forget, output and cell-state gate, respectively and Eistein's summation convention is employed, i.e. indices that appear twice are summed over. The calligraphic symbols $\mathcal{M}_{hk}^g, \mathcal{N}_{hh'}^g$ and $\mathcal{B}_h^g$ are the free parameters of the network that are optimized for the given problem during the training, which is at the heart of the any machine learning approach. The matrix $\boldsymbol{\mathcal{M}}^g = (\mathcal{M}_{hk}^g) \in \mathbb{R}^{N_h \times N_k}$ can be interpreted as a Markovian-like contribution of the current input $\vec{X}^t[\tau]$ to the gates, whereas the $\boldsymbol{\mathcal{N}}^g = (\mathcal{N}_{hh'}^g) \in \mathbb{R}^{N_h \times N_h}$ scales a non-Markovian part determined by the hidden state of the last sequence point $\tau - 1$. The vector $\vec{\mathcal{B}}^g = (\mathcal{B}_h^g) \in \mathbb{R}^{N_h}$ is a learnable bias. It should be stressed that these parameters do neither depend on t nor on τ and are thus optimized once for the complete dataset $\{\vec{X}^t\}$.

Note that this mapping is sometimes extended by a contribution to the $g_h^t[\tau]$ from the past cell state $\vec{C}^t[\tau - 1]$ (**XXX?**). Nevertheless, this mechanism called "peeping" is not further considered in this work.

For the sake of brevity, we can write the mapping more compactly in matrix-vector form as

$$\vec{g}^t[\tau] = \boldsymbol{\mathcal{M}}^g \vec{X}^t[\tau] + \boldsymbol{\mathcal{N}}^g \vec{H}^t[\tau - 1] + \vec{\mathcal{B}}^g \tag{1}$$

Second, the actual gate vectors are computed by the core equations of the LSTM as proposed by Hochreiter & Schmidhuber (1997):

$$\begin{aligned}
\vec{I}^t[\tau] &= \sigma(\vec{i}^t[\tau]) \\
\vec{F}^t[\tau] &= \sigma(\vec{f}^t[\tau]) \\
\vec{O}^t[\tau] &= \sigma(\vec{o}^t[\tau]) \\
\vec{C}^t[\tau] &= \vec{F}^t[\tau] \circ \vec{C}^t[\tau - 1] + \vec{I}^t[\tau] \circ \tanh(\vec{c}^t[\tau]) \\
\vec{H}^t[\tau] &= \vec{O}^t[\tau] \circ \tanh(\vec{C}^t[\tau]) \ ,
\end{aligned} \tag{2}$$

where σ denotes the logisitc sigmoid function, tanh is the hyperbolic tangent and the $\circ$ stands for the Hadamard product (all applied in an element-wise fashion to the vectors). In the last two equations the role of the input, forget and output gates as described above becomes apparent.

The third step in a single layer LSTM (as employed for the work presented here) is then to provide the output of the current LSTM cell, i.e. $\vec{H}^t[\tau]$ and $\vec{C}^t[\tau]$, to the subsequent LSTM cell that processes the next element $\vec{X}^t[\tau+1]$ of the input sequence.

The full action of the LSTM network up to the end of the sequence can be written as a nested function call

$$\left(\vec{H}^t[N_\tau], \vec{C}^t[N_\tau]\right) = L\left(\vec{X}^t[N_\tau], L\left(\vec{X}^t[N_\tau - 1], \dots L\left(\vec{X}^t[1], (\vec{H}^t[0], \vec{C}^t[0])\right) \dots\right)\right) \ , \quad (3)$$

where $L\left(\vec{X}^t[\tau], \left(\vec{H}^t[\tau-1], \vec{C}^t[\tau-1]\right)\right)$ represents Equation 1 and Equation 2. For the present work, the initial conditions are chosen as $\vec{H}^t[0] = \vec{C}^t[0] = 0$, which simply means that there is no memory longer than N_τ time steps.

The final output of the ConvLSTM chain, $\vec{H}^t[N_\tau]$ and $\vec{C}^t[N_\tau]$, encode information on the full input sequence ending at time t. This information has to be decoded to obtain useful information via an appropriate subsequent network, as it is described in the Section 2.3.

2.2.2 Combining LSTM with spatial convolution

Although the plain LSTM has high performance in handling temporal sequences of point-like quantities it is not designed to recognize spatial features in seuqences of, e.g., two-dimensional maps as atmospheric-ocean interface fields. To adress this limitation we employ a ConvLSTM architecture as described in the following.

In this kind of network the elements of the input sequence are given as spatially varying fields $\vec{X}^t[\tau] = (X^t_k[x, y, \tau]) \in \mathbb{R}^{N_k \times (N_x \times N_y)}$, where $x \in [1, N_x]$ and $y \in [1, N_y]$ run over the horizontal and vertical dimensions of the map, respectively. In order to enable the "learning" of spatial patterns, the free parameters of the network are replaced by two-dimenmsional convolutional kernels $\boldsymbol{\mathcal{M}}^g = (\mathcal{M}^g_{hk}[\xi, \eta]) \in \mathbb{R}^{(N_h \times N_k) \times (N_\xi \times N_\eta)}$ and $\boldsymbol{\mathcal{N}}^g = (\mathcal{N}^g_{hh'}[\xi, \eta]) \in \mathbb{R}^{(N_h \times N_h) \times (N_\xi \times N_\eta)}$. The width and the height of the kernels are given by N_ξ and N_η, respectively and $\xi \in [-(N_\xi - 1)/2, (N_\xi - 1)/2], \eta \in [-(N_\eta - 1)/2, (N_\eta - 1)/2]$, where, without loss of generality, we assume odd numbers for the kernel sizes.

The mapping from the input quantities to the gates is then given by a convolution with these kernels

$$g^t_h[x, y, \tau] = \mathcal{M}^g_{hk}[\xi, \eta]\, X^t_k[x - \xi, y - \eta, \tau] + \mathcal{N}^g_{hh'}[\xi, \eta]\, H^t_{h'}[x - \xi, y - \eta, \tau - 1] + \mathcal{B}^g_h \ .$$

again with Einstein's convention imposed.

It becomes immediately apparent that in case of the ConvLSTM, the gate and state vectors must become vector fields ($\in \mathbb{R}^{N_h \times (N_x \times N_y)}$) as well. We can write this mapping in the same fashion as Equation 1 but with replacing the normal matrix-vector multiplication by a convolution (denoted with $*$), i.e.

$$\vec{g}^t[\tau] = \boldsymbol{\mathcal{M}}^g * \vec{X}^t[\tau] + \boldsymbol{\mathcal{N}}^g * \vec{H}^t[\tau - 1] + \vec{\mathcal{B}}^g$$

The susebsequent processing of the $\vec{g}^t[\tau]$ remains symbolically the same as presented in Equation 2 but with all appearing quantities now meaning vector fields instead of simple vectors.

In summary, the ConvLSTM excels at processing tasks that demand a combined understanding of spatial patterns and temporal sequences in data. It merges the image-processing capabilities of Convolutional Neural Networks (CNNs) with the time-series modeling of Long Short-Term Memory (LSTM) networks.

2.3 Fully connected layer

As stated in Sec. Section 2.2.1, the final output $\vec{H}^t[N_\tau]$ and $\vec{C}^t[N_\tau]$ of the ConvLSTM encode information on the full input sequence. In order to contract this information to obtain the runoff vector $\vec{R}^t$ representing the N_r rivers, we propose to subject the final short-term memory (i.e. the hidden state $\vec{H}^t[N_\tau]$) to an additional FC network.

In particular, the dimensionality of the vector field $\vec{H}^t[N_\tau]$ is sequentially reduced by three nested FC layers, each connected to the other by the Rectified Linear Unit (ReLU), see Figure 1. Integrating over artificial degrees of freedom in a step-wise fashion has turned out to be benificial [(**XXX?**)].

Spelled out in mathematics, the runoff of the r-th river is then obtained via (using Einstein's convention)

$$R_r^t = \mathcal{W}_{rb}^3 \text{ReLU}\left(\mathcal{W}_{ba}^2 \text{ReLU}\left(\mathcal{W}_{ah}^1[x,y] H_h^t[x,y,N_\tau] + \mathcal{B}_a^1\right) + \mathcal{B}_b^2\right) + \mathcal{B}_r^3 \ ,$$

where $a = 1, \dots N_a$, $b = 1, \dots N_b$ and the hyper parameters N_a and N_b are chosen such that $N_h \cdot N_x \cdot N_y > N_a > N_b > N_r$ in order to achieve the aforementioned step-by-step reduction of dimensionality. The weights $\mathcal{W}$ and biases $\mathcal{B}$ stand for parameters that are optimized during the training of the network.

In matrix-vector notation this can be compactified to

$$\vec{R}^t = \boldsymbol{\mathcal{W}}^3 \text{ReLU}\left(\boldsymbol{\mathcal{W}}^2 \text{ReLU}\left(\boldsymbol{\mathcal{W}}^1 \vec{H}^t[N_\tau] + \vec{\mathcal{B}}^1\right) + \vec{\mathcal{B}}^2\right) + \vec{\mathcal{B}}^3 \ .$$

Combining the last equation with Equation 3 provides finally an explicit formula for the initial assumption of modelling the runoff for time t as a functional of a sequence of atmospheric fields, i.e.

$$\begin{aligned}\vec{R}^t &= \vec{M}(\{X_k^t[x,y,\tau]\}) \\ &= \boldsymbol{\mathcal{W}}^3 \text{ReLU}\left(\boldsymbol{\mathcal{W}}^2 \text{ReLU}\left(\boldsymbol{\mathcal{W}}^1 \vec{L}_H\left(\vec{X}^t[N_\tau], L\left(\vec{X}^t[N_\tau - 1], \dots L\left(\vec{X}^t[1], (0,0)\right) \dots\right)\right) + \vec{\mathcal{B}}^1\right) + \vec{\mathcal{B}}^2\right) + \vec{\mathcal{B}}^3 \ ,\end{aligned}$$

where the $\vec{L}_H$ means that only the hidden state vector of the final ConvLSTM call is forwarded to the FC layer.

In Section 3, we present the employed model and data setup as well as the choice for all mentioned hyper parameters that lead to an adequate model performance after training.

3 Technical details

3.1 Runoff data used for training

The runoff data covering the period 1979 to 2011 is based on an E-HYPE hindcast simulation that was forced by a regional downscaling of ERA-Interim (Dee et al., 2011) with RCA3 ("The rossby centre regional climate model RCA3," n.d.) and implemented into NEMO-Nordic (Hordoir et al., 2019) as a mass flux. For the periods before (1961 to 1978) and after (2012 to 2018) additional spatial temporal corrections have been applied to the runoff data, and have therefore been ignored. For more information see Gröger et al. (2022) and references therein.

3.2 Atmospheric Forcing

The UERRA-HARMONIE regional reanalysis dataset was developed as part of the FP7 UERRA project (Uncertainties in Ensembles of Regional Re-Analyses, http://www.uerra.eu/),). The UERRA-HARMONIE system represents a comprehensive, high-resolution reanalysis covering a wide range of essential climate variables. This dataset encompasses data on air temperature, pressure, humidity, wind speed and direction, cloud cover, precipitation, albedo, surface heat fluxes, and radiation fluxes from January 1961 to July 2019. With a horizontal resolution of 11 km and analyses conducted at 00 UTC, 06 UTC, 12 UTC, and 18 UTC, it also provides hourly resolution forecast model data. UERRA-HARMONIE is accessible through the Copernicus Climate Data Store (CDS, https://cds.climate.copernicus.eu/#!/home), initially produced during the UERRA project and later transitioned to the Copernicus Climate Change Service (C3S, https://climate.copernicus.eu/copernicus-regionalreanalysis-europe).

3.3 Ocean Model

To simulate the Baltic Sea, we use a coupled 3-dimensional ocean model Baltic Sea, called the Modular Ocean Model (MOM). This model uses a finite-difference method to solve the full set of primitive equations to calculate the motion of water and the transport of heat and salt. The K-profile parameterization (KPP) was used as turbulence closure scheme. The model's western boundary opens into the Skagerrak and connects the Baltic Sea to the North Sea. The maximum depth was set at 264 meters. A more detailed description of the setup can be found in (Gröger et al., 2022).

3.4 Neural network hyper parameters

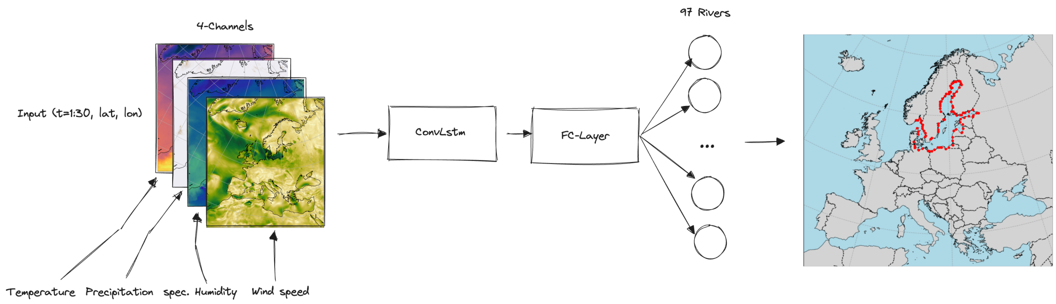


Figure 3: Schematic structure of the ConvLSTM implementation for river runoff forecasting.

For the computation we use the following set of hyper parameters:

Table 1: Hyperparameters

Parameter name	Parameter size
Channel size	4
Num. hidden layer	9
Num. timesteps	30
Conv. Kernelsize	(7,7)
Num. ConvLSTM layers	1
Batch size	50
Learning Rate	1e-3 with CosineAnnealing

As input the model receives 30 days of atmospheric surface fields temperature T, precipitation P, specific humidity Q and wind speed W, with a daily resolution to predict the river runoff R at the time step $\Delta t + 1$, which can be summarized as

$$R_{\Delta t+1} = f\left(T_{t-30:t}, P_{t-30:t}, Q_{t-30:t}, W_{t-30:t}\right)$$

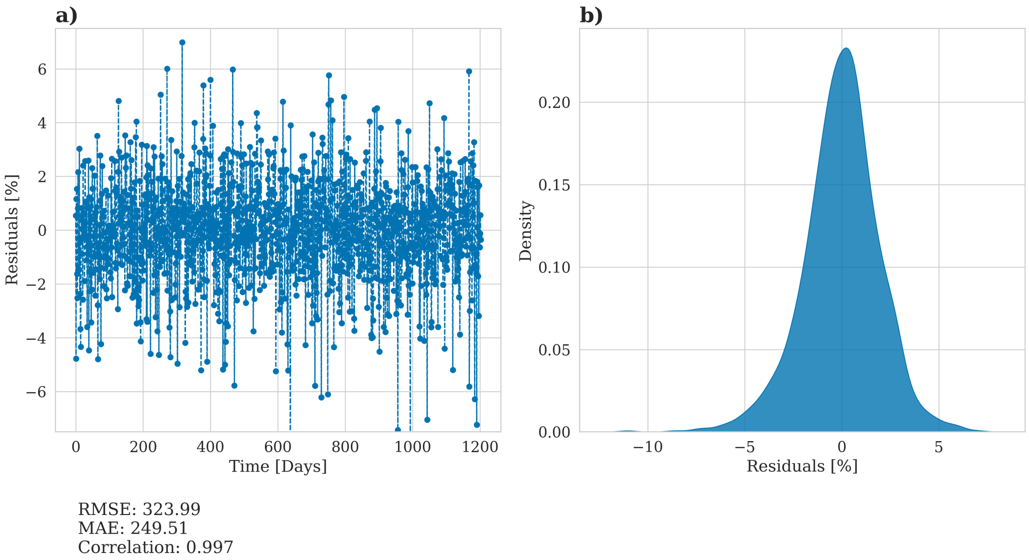
with f being a function maps the 30 days of daily atmospheric surface fields data to the predicted river runoff.

The choice of atmospheric fields was based on the implemented river runoff calculation in the atmospheric model COSMO-CLM which uses these four fields to calculate an river runoff estimate.

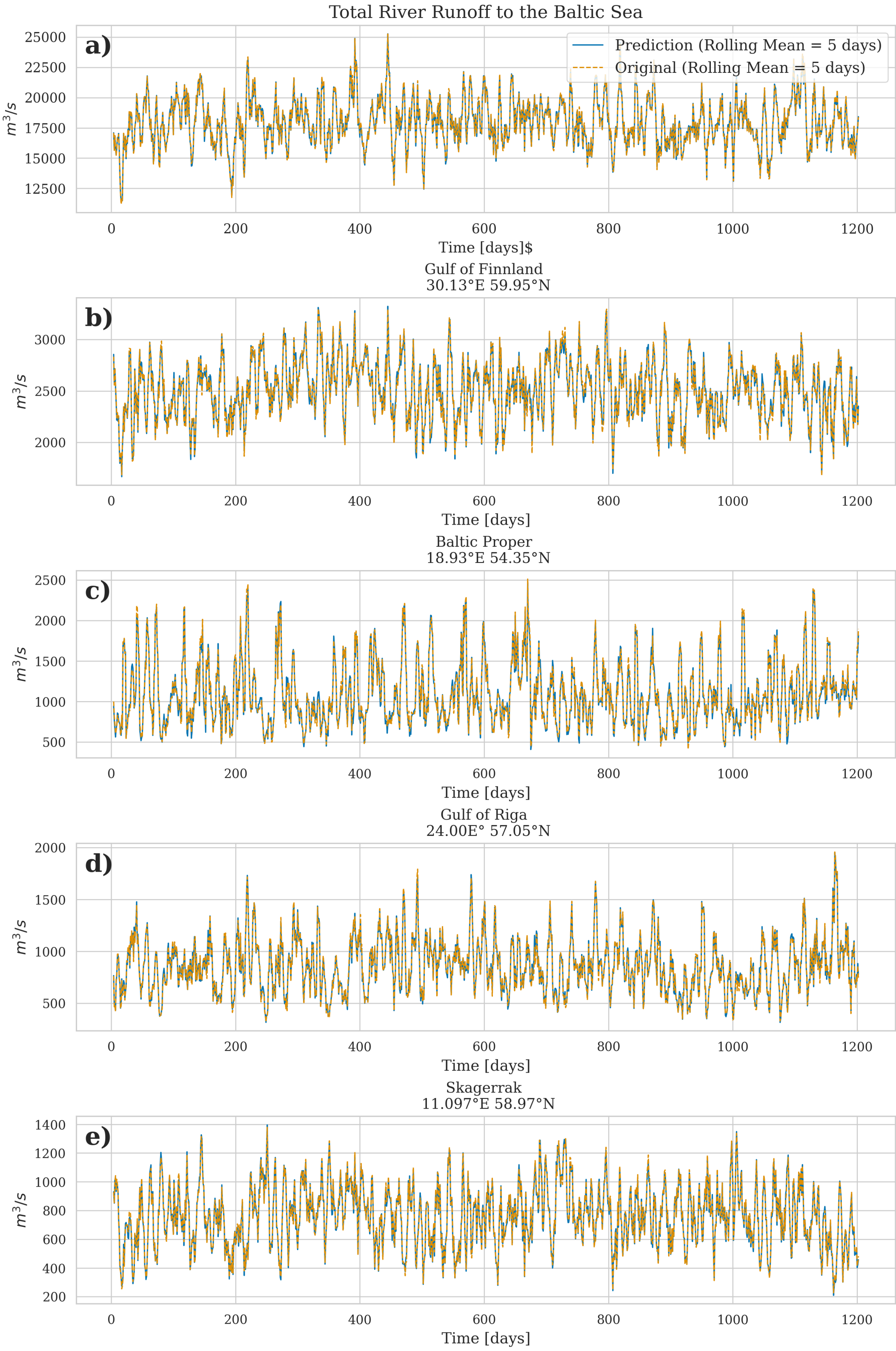
4 Results

The model was trained with daily data for the period 1979 to 2011, as this period represents the only period of E-HYPE without further bias correction applied to the runoff to match observations. The data was divided into randomly chosen splits of 80% training data, 10% validation data to evaluate the performance of the model during training, and 10% training data which is finally used to evaluate the performance of the model after training. The model was trained for 400 epochs and the model weights with the lowest mean squared error during training have been stored.

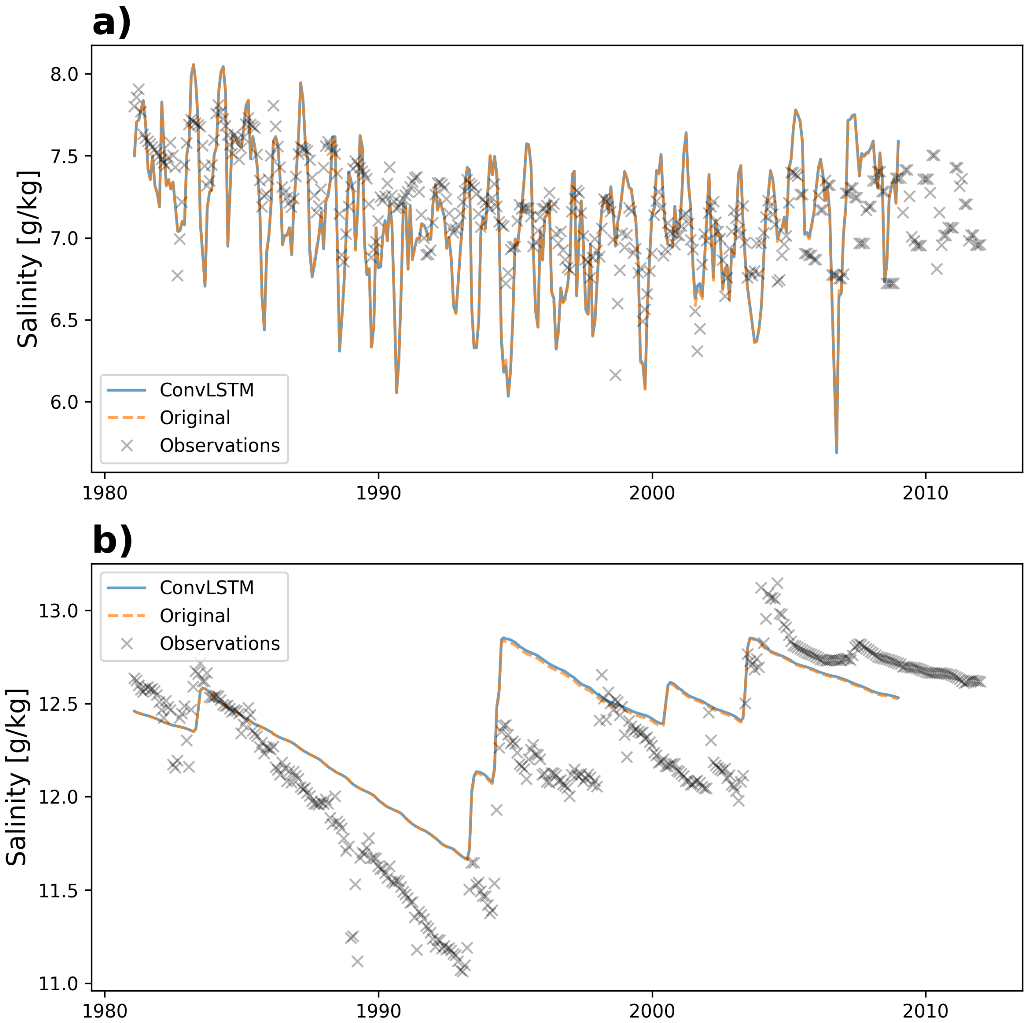
The accuracy of the model is displayed in Figure **?@fig-statistical-evaluationNN**. As mention above for evaluation, the test dataset was utilized. The left panel panel (Figure **?@fig-statistical-evaluationNN** a)) illustrates the relative prediction error in relation to the original E-HYPE data, indicating that, on daily timescales, the model can predict river runoff with an accuracy of $\pm$ 5%. The overall correlation is 0.997 with the resulting error metrics yielding a RMSE of 323.99 m^3/s and MAE of 249.51 m^3/s. While the model's performance is satisfactory, the discrepancies between the actual values and the predictions could partly be attributed to the use of a different atmospheric dataset than the one originally used to drive the E-HYPE model. However, by applying a rolling mean with a 5-day window, the prediction error is reduced to less than 1%, which is acceptable for the purposes of climate modeling. Lastly, the right panel demonstrates that the distribution of residuals follows a Gaussian shape, suggesting that our model does not exhibit bias by systematically over or underestimating river runoff values.



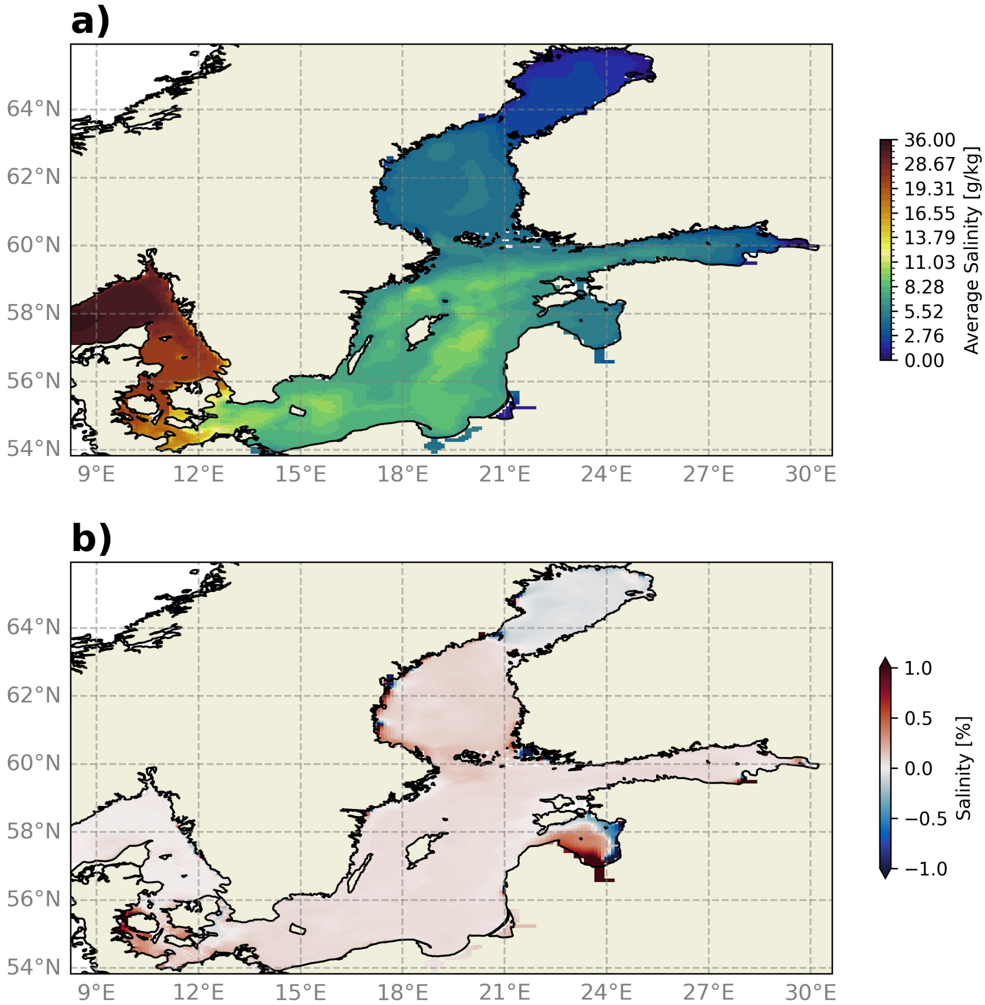
In the following we will now address the overall performance of the total river runoff while also zooming in on the four largest rivers entering the Baltic Sea. @fig-PerformanceNeuralNetworkRunoff shows the predicted and the original river runoff using the test dataset. The predicted total river runoff for the Baltic Sea is closely matching the original data (@fig-PerformanceNeuralNetworkRunoff a). Zooming in on the largest individual rivers (@fig-PerformanceNeuralNetworkRunoff b-e) it can be seen that that also the prediction of the individual rivers is close to the original data.



Lastly, we evaluated the performance of the runoff model by incorporating the predicted river runoff as runoff forcing into the ocean model MOM5. This provides a robust validation of the runoff model against more complex real world conditions. This allows us to ensure that the predictions accurately reflect the impact of the river discharge on the ocean dynamics, validating the temporal and spatial variability of the the river discharge. **?@fig-by15** shows the salinity comparison between the original E-HYPE river runoff and the predicted river runoff at BY15 - a central stations in the Baltic Sea. It can be seen that the model simulation using the predicted river runoff by the ConvLSTM is closely mirroring the control simulation. The upper panel (**?@fig-by15** a) shows the surface salinity, representing the high-frequency variations in the salinity, which is heavily affected by river runoff. The lower panel (**?@fig-by15** b) shows the bottom salinity which can be viewed as a low-pass filter in the Baltic Sea, which is also closely mirrored by the ConvLSTM predictions.



The final evaluation of the ConvLSTM model concentrates on the spatial accuracy of river runoff predictions. Figure xxx a) shows the vertically averaged salinity for the period 1981 to 2011 in the reference simulation. It highlights the strong horizontal gradients and complex topographic features in the Baltic Sea, as evidenced by salinity variations in deeper waters, captured by the vertical integration. Figure 4b compares these results by showing the percentage difference in vertically averaged salinity between the ConvLSTM simulation and the reference simulation. Overall, the differences remain below 1%, except near a river mouth in the Gulf of Riga (22-24°E, 56.5-58.5°N for orientation), where the difference is approximately 1%.



5 Discussion

With the increasing demand of decision makers for regional climate projections, that allow to quantify regional climate change impacts, the availability of precise hydrological forecasting becomes invaluable. The quality of a projection for a coastal sea such as the Baltic Sea is too large parts based on the quality of the hyrdological conditions. In this work we analyze the implementation of Convolutional Long Short-Term Memory (ConvLSTM) networks for predicting river runoff, highlighting its potential to enhance river runoff forecasting across different coastal seas. Given the unique hydrological characteristics of the Baltic Sea, largely influenced by its limited connection to the open ocean and significant freshwater input from surrounding rivers, the region presents a critical case for the application of sophisticated forecasting models.

The transition from traditional hydrological models to machine learning approaches, such as the ConvLSTM model, offers significant advantages. The ConvLSTM can be seamlessly integrated into regional climate models, allowing for the real-time computation of river runoff while performing climate projections. While the initial training of the model requires substantial computational resources, it remains less intensive than running comprehensive hydrological models. Furthermore, once trained, the ConvLSTM model is computationally efficient during inference, ensuring enhanced forecasting capabilities without significantly increasing computational demands.

While the ConvLSTM represents an advancement for the climate community, given that running and tuning traditional hydrological models demands extensive expertise, models like E-HYPE maintain an essential role. They provide a comprehensive dataset that helps to train our ConvLSTM model effectively. This robust training enables the machine learning models to achieve highly accurate predictive weights. Thus, rather than rendering traditional methods obsolete, the integration of machine learning models builds upon and enhances the foundational data provided by them.

The robust performance of the ConvLSTM model in simulating river runoff and its possible effective integration into regional climate models pave the way for a multitude of new storyline simulations. Hence, we can now explore various "what-if" scenarios more reliably, under the assumption that the model weights attained during training are robust.

In summary, we showed that the ConvLSTM model demonstrated robust performance in forecasting river runoff across 97 rivers entering the Baltic Sea. Trained on data from 1979 to 2011, the model achieved an impressive daily prediction accuracy of $\pm 5\%$. This capability to generate accurate and detailed simulations enables to examine the potential impacts of different climate scenarios. Such precision in forecasting and scenario testing is crucial for crafting effective water resource management strategies and adapting to the changing climate.

Ultimately, the integration of ConvLSTM into regional climate models represents a significant step forward in our ability to understand and predict the complex dynamics of river systems and their impact on regional climate systems.

6 Acknowledgments

Phasellus interdum tincidunt ex, a euismod massa pulvinar at. Ut fringilla ut nisi nec volutpat. Morbi imperdiet congue tincidunt. Vivamus eget rutrum purus. Etiam et pretium justo. Donec et egestas sem. Donec molestie ex sit amet viverra egestas. Nullam justo nulla, fringilla at iaculis in, posuere non mauris. Ut eget imperdiet elit.

7 Open research

Phasellus interdum tincidunt ex, a euismod massa pulvinar at. Ut fringilla ut nisi nec volutpat. Morbi imperdiet congue tincidunt. Vivamus eget rutrum purus. Etiam et pretium justo. Donec et egestas sem. Donec molestie ex sit amet viverra egestas. Nullam justo nulla, fringilla at iaculis in, posuere non mauris. Ut eget imperdiet elit.

References

Ashrafi, M., Chua, L. H. C., Quek, C., & Qin, X. (2017). A fully-online neuro-fuzzy model for flow forecasting in basins with limited data. *Journal of Hydrology*, *545*, 424–435.

Dee, D. P., Uppala, S. M., Simmons, A. J., Berrisford, P., Poli, P., Kobayashi, S., et al. (2011). The ERA-Interim reanalysis: configuration and performance of the data assimilation system. *Quarterly Journal of the Royal Meteorological Society*, *137*(656), 553–597. https://doi.org/10.1002/qj.828

Fang, L., & Shao, D. (2022). Application of long short-term memory (LSTM) on the prediction of rainfall-runoff in karst area. *Frontiers in Physics*, *9*, 790687.

Fang, W., Huang, S., Ren, K., Huang, Q., Huang, G., Cheng, G., & Li, K. (2019). Examining the applicability of different sampling techniques in the development of decomposition-based streamflow forecasting models. *Journal of Hydrology*, *568*, 534–550. https://doi.org/10.1016/j.jhydrol.2018.11.020

Gröger, M., Placke, M., Meier, M., Börgel, F., Brunnabend, S.-E., Dutheil, C., et al. (2022). *The Baltic Sea model inter-comparison project BMIP – a platform for model development, evaluation, and uncertainty assessment.* Retrieved from https://gmd.copernicus.org/preprints/gmd-2022-160/

Ha, S., Liu, D., & Mu, L. (2021). Prediction of Yangtze River streamflow based on deep learning neural network with El Niño–Southern Oscillation. *Scientific Reports*, *11*(1), 11738. https://doi.org/10.1038/s41598-021-90964-3

Hagemann, S., Stacke, T., & Ho-Hagemann, H. T. M. (2020). High Resolution Discharge Simulations Over Europe and the Baltic Sea Catchment. *Frontiers in Earth Science*, *8*. https://doi.org/10.3389/feart.2020.00012

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780.

Hordoir, R., Axell, L., Höglund, A., Dieterich, C., Fransner, F., Gröger, M., et al. (2019). Nemo-nordic 1.0: A NEMO-based ocean model for the baltic and north seas – research and operational applications. *Geoscientific Model Development*, *12*(1), 363–386. https://doi.org/10.5194/gmd-12-363-2019

Huang, S., Chang, J., Huang, Q., & Chen, Y. (2014). Monthly streamflow prediction using modified EMD-based support vector machine. *Journal of Hydrology*, *511*, 764–775.

Humphrey, G. B., Gibbs, M. S., Dandy, G. C., & Maier, H. R. (2016). A hybrid approach to monthly streamflow forecasting: Integrating hydrological model outputs into a bayesian artificial neural network. *Journal of Hydrology*, *540*, 623–640.

Kratzert, F., Klotz, D., Brenner, C., Schulz, K., & Herrnegger, M. (2018). Rainfall–runoff modelling using long short-term memory (LSTM) networks. *Hydrology and Earth System Sciences*, *22*(11), 6005–6022.

Liu, D., Jiang, W., Mu, L., & Wang, S. (2020). Streamflow prediction using deep learning neural network: Case study of yangtze river. *IEEE Access*, *8*, 90069–90086.

Meier, H. M., & Döscher, R. (2002). Simulated water and heat cycles of the baltic sea using a 3D coupled atmosphere–ice–ocean model. *Boreal Environment Research*, *7*(4), 327.

Moishin, M., Deo, R. C., Prasad, R., Raj, N., & Abdulla, S. (2021). Designing deep-based learning flood forecast model with ConvLSTM hybrid algorithm. *IEEE Access*, *9*, 50982–50993.

SHI, X., Chen, Z., Wang, H., Yeung, D.-Y., Wong, W., & WOO, W. (2015). Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting. In *Advances in Neural Information Processing Systems* (Vol. 28). Curran Associates, Inc. Retrieved from https://proceedings.neurips.cc/paper/2015/hash/07563a3fe3bbe7e3ba84431ad9d055af-Abstract.html

Tan, Q.-F., Lei, X.-H., Wang, X., Wang, H., Wen, X., Ji, Y., & Kang, A.-Q. (2018). An adaptive middle and long-term runoff forecast model using EEMD-ANN hybrid approach. *Journal of Hydrology*, *567*, 767–780. https://doi.org/10.1016/j.jhydrol.2018.01.015

The rossby centre regional climate model RCA3: Model description and performance - tellus a: Dynamic meteorology and oceanography. (n.d.). Retrieved from https://a.tellusjournals.se/articles/10.1111/j.1600-0870.2010.00478.x

Zhu, S., Wei, J., Zhang, H., Xu, Y., & Qin, H. (2023). Spatiotemporal deep learning rainfall-runoff forecasting combined with remote sensing precipitation products in large scale basins. *Journal of Hydrology*, *616*, 128727.