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 hyrdological 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 intergrate statistical correction, using the land surface schemes of global or regional climate models.

The releatively recent rise of machine learning (ML) models has been mostly explored for river runoff forecasting, as accurate runoff forecasting, especially over extended periods, is pivotal for effective water resources management (Fang et al., 2019; Tan et al., 2018; Yang et al., 2018). Common approaches employ 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 (Humphrey et al 2016, Huang et al 2014, Ashrafi et al 2017, Yuan et al 2018, Xu et al 2021).

LSTM networks, an evolution of the classical Recurrent Neural Networks (RNNs), have shown stability and efficacy in sequence-to-sequence predictions, such as using climatic indices for rainfall estimation or long-term hydrological forecasting. However, a limitation of LSTMs is their inability to effectively capture two-dimensional structures, an area where Convolutional Neural Networks (CNNs) excel. Reconizing this limitation (SHI et al., 2015) proposed a convolutional LSTM (ConvLSTM) architectures, which combines the strengths of both LSTM and CNN. The ConvLSTM network has been proven useful for precipitation nowcasting (SHI et al., 2015) or for river runoff forecasting [Ha et al. (2021); @niu2020].

In the following we will show that in absence of a fully functioning hydrogolocial model, that also uses a rather complex parametrization, ConvLSTM also represent a robust way to predict multiple rivers at once for any given period using only atmospheric forcing. In this work we use the Baltic Sea catchment to illustrate our approach, while in principal the methodology we propose is universally applicable across various geographic regions. The Baltic Sea serves as a challenging example 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. More generally, the freshwater input into the Baltic Sea comes either as river runoff or a 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 and Doescher, 2002). Modeling 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.

2 Methods

2.1 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.

2.2 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).

2.3 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).

2.4 LSTM network

Before turning 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 $\vec{X}^t[1], \ldots \vec{X}^t[\tau], \ldots \vec{X}^t[N_\tau]$ 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. $\vec{X}^t[N_\tau] \equiv \vec{X}^t$. 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 1).

This performance in modeling long-range dependencies has been validated in various studies. The key component of LSTM's innovation is its memory cell, $\vec{C}^t[\tau] = (C_h^t[\tau]) \in \mathbb{R}^{N_h}$, which stores state information, also referred to as long-term memory. The index $\tau \in [1, N_\tau]$ corresponds to the τ-th element in the input

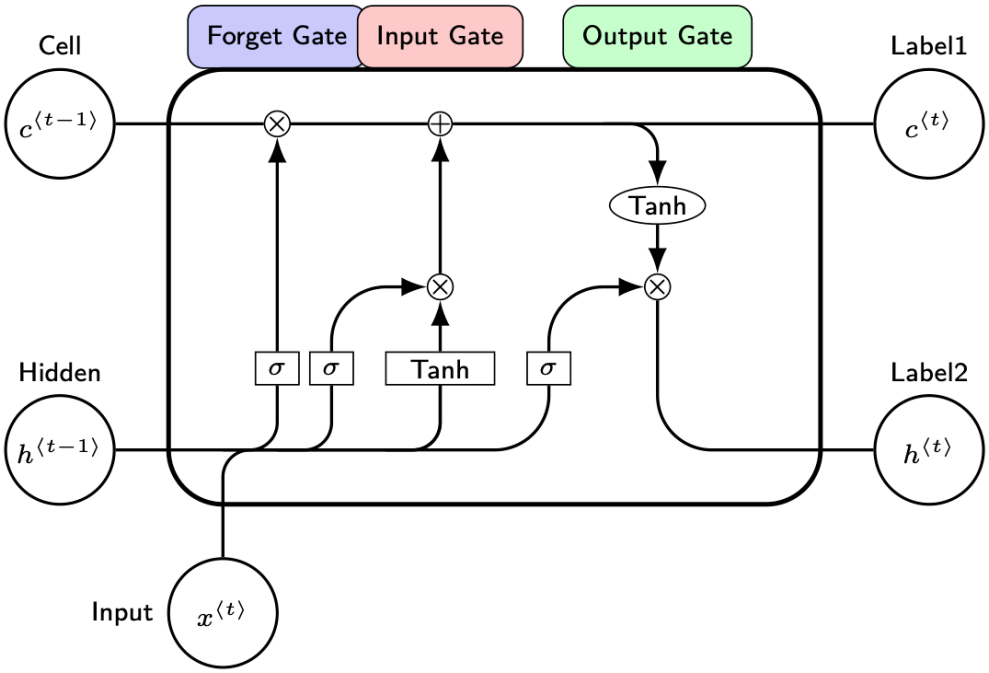


Figure 1: Inner structure of a Long Short-Term Memory Cell

sequence. The cell state is a vector consists of elements, each associated with one of the N_h hidden layers, labeled by h. This vector is determined through several self-parameterized gates, all in the same vector space as $\vec{C}^t[\tau]$. The forget gate $\vec{F}^t[\tau]$ defines the percentage of the previous long-term memory status $\vec{C}^t[\tau-1]$ that should be retained stored. The input gate $\vec{I}^t[\tau]$ decides how much of the input is added to the the long-term memory, forming the updated cell state. The so-called output gate $\vec{O}^t[\tau]$ then determines how much of the latest cell output, $\vec{C}^t[\tau]$, is propagated to the final state, $\vec{H}^t[\tau]$ representing the updated short-term memory of the hidden state.

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 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 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 (**XXX?**):

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

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 become 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 therefore be written as a nested function call

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

where L represents Eqs. (Equation 1, Equation 2) and the initial conditions are usually chosen as $\vec{H}^t[0] = \vec{C}^t[0] = 0$.

The final output $\vec{H}^t[N_\tau]$ 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 usueful information via an appropriate subsequent layer, as it is described in the Sec. (**XXX?**).

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

2.5 ConvLSTM network

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 two-dimensional maps. To adress this limitation we use a convLSTM architecture.

In this kind of network the elements of the input sequence are given as (kind of) vector fields $\vec{X}^t[\tau] = (X_k^t[x, y, \tau]) \in \mathbb{R}^{N_k \times (N_x \times N_y)}$, 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 $\boldsymbol{\mathcal{M}}^g = (\mathcal{M}_{hk}^g[\xi, \eta]) \in \mathbb{R}^{(N_h \times N_k) \times (N_\xi \times N_\eta)}$ and $\boldsymbol{\mathcal{N}}^g = (\mathcal{N}_{hh'}^g[\xi, \eta]) \in \mathbb{R}^{(N_h \times N_h) \times (N_\xi \times N_\eta)}$.

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 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

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

It becomes immediatly apparent that in case of the convLSTM, the bias, gate and state vectors must become vector fields ($\in \quad \mathbb{R}^{N_h \times (N_x \times N_y)}$) as well. We can write this mapping in the same fashion as Eq. (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 Eq. (Equation 2) but with all appearing quantities meaning standing now for 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.

2.6 Implemented model architecture

The ConvLSTM architectures uses and encoder/decoder structure as discussed in TODO. To predict all 97 rivers entering the Baltic Sea, we flatten the output and use fully connected layers to map onto the individual rivers outputs.

An overview of the model structure is given below

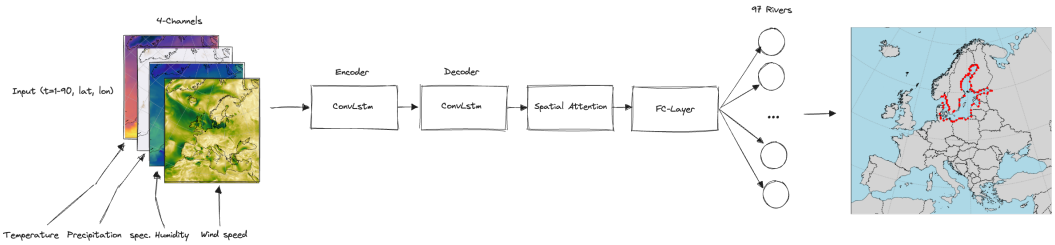


Figure 2: BaltConvLSTM

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

Table 1: Hyperparameters

Parameter name	Parameter size
Channel size	4
Num. hidden layer	10
Num. timesteps	30
Conv. Kernelsize	(9,9)
Num. ConvLSTM layers	1
Batch size	64
Learning Rate	1e-3 with CosineAnnealing

Parameter name	Parameter size

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

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

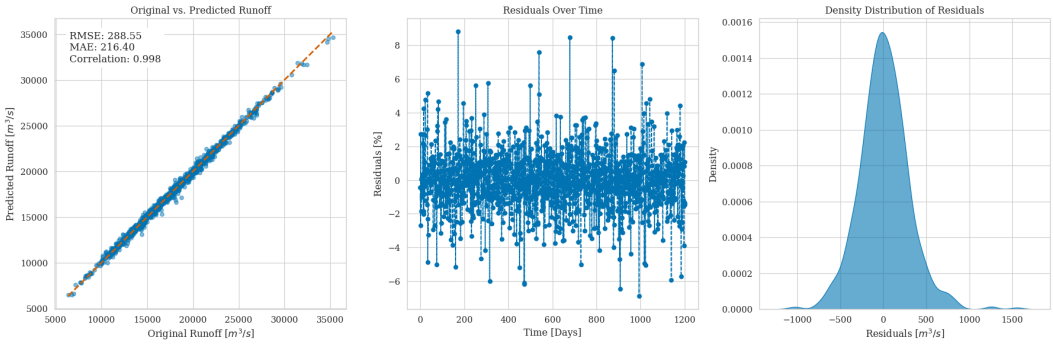
with f being a function maps the 30 days of daily atmospheric surface fields data to the predicted river runoff.

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.

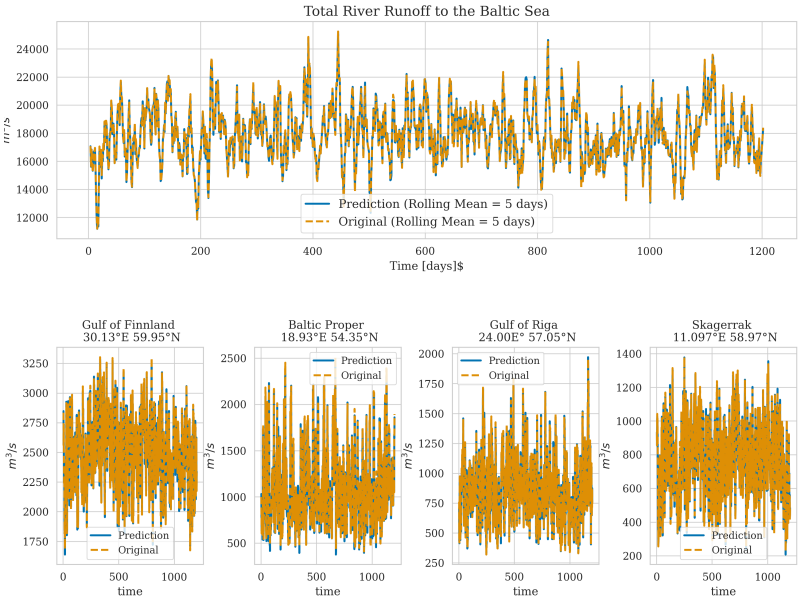
3 Results

For the evaluation of the model performance we consider the period 1979 to 2011. For this period no bias correction was applied to the orignal E-HYPE dataset. We chose a split of 80% training data, 10% validation data to evaluate the performance of the model during training, and 10% training data that is finally used to evaluate the performance of the model after training.

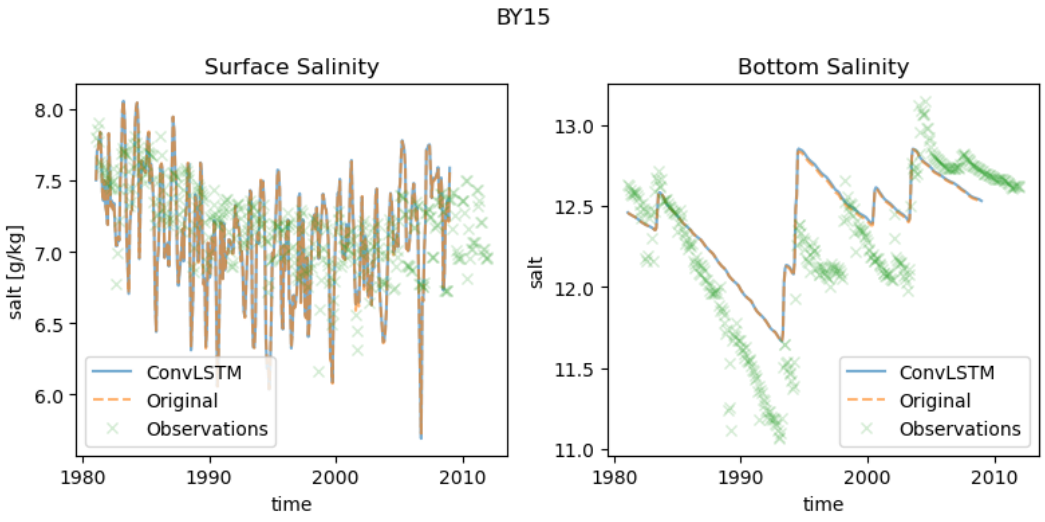
The accuracy of the model is further displayed in **?@fig-statistical-evaluationNN** . The correlation is close to 1 and the residuals show the error of the model is below 1% of the original data.

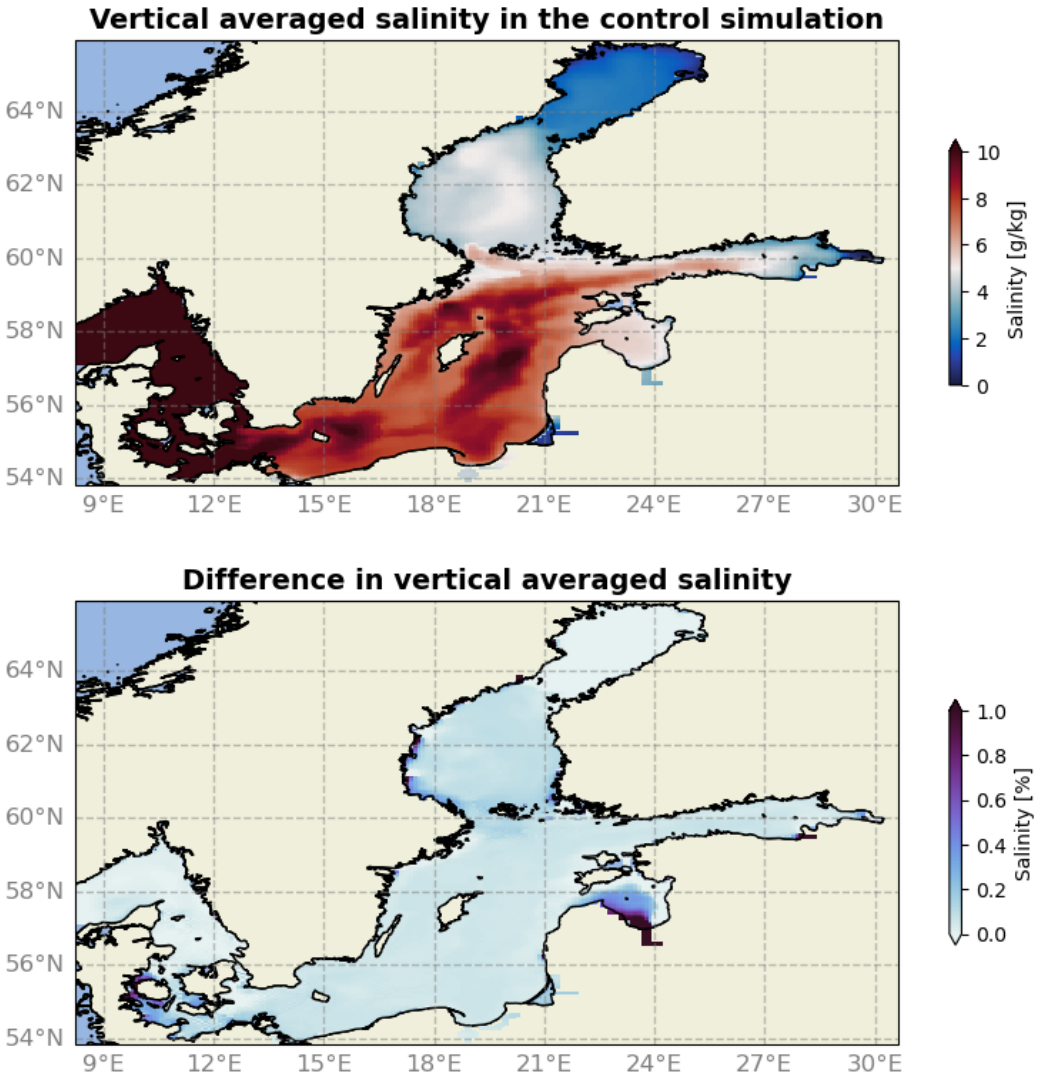


?@fig-PerformanceNeuralNetworkRunoff shows the performance of the model using the test dataset. The predicted total river runoff for the Baltic Sea is closely matching the original data. Zooming in on the largest individual rivers (lower panels) it can be seen that that also the prediction of the inidivual rivers is close to the original data.



Lastly, we evaluated the performance of the runoff model by incorporating the predicted river runoff as 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.





4 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.

5 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

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, 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

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

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 (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

Yang, Z., Liu, P., Cheng, L., Wang, H., Ming, B., & Gong, W. (2018). Deriving operating rules for a large-scale hydro-photovoltaic power system using implicit stochastic optimization. *Journal of Cleaner Production*, *195*, 562–572. https://doi.org/10.1016/j.jclepro.2018.05.154