Rainfall–runoff modeling using Long Short-Term Memory (LSTM) networks: Exploring the Capabilities of LSTM Cell States to Model Physical Processes in Hydrological Forecasting

Jouval Somer[1], Philip Hoel[2]
[1]Department of Mathematics, University of Oslo, Norway
[2]Department of Informatics, University of Oslo, Norway

This study explores the application of Long Short-Term Memory (LSTM) networks in rainfall-runoff modeling across diverse hydrological settings in Norway. Given the challenges imposed by traditional hydrological models—particularly under extreme weather conditions and varied land uses—this research integrates LSTM to leverage its advanced pattern recognition capabilities and handle sequential data efficiently. The focus is on examining the LSTM's effectiveness in capturing the dynamics of Snow Water Equivalent (SWE) and other meteorological features crucial for accurate hydrological forecasting. By employing LSTMs, we aim to enhance model reliability and provide a nuanced understanding of hydrological processes, potentially overcoming the limitations of existing models. The findings suggest that LSTMs can significantly improve prediction accuracy, demonstrating robustness in environments where traditional models falter due to limited data availability or complex hydrological dynamics. This work underscores the potential of machine learning techniques in the enhancement of hydrological forecasting and suggests pathways for future research in the integration of such technologies in environmental science.

Index Terms—**Hydrological modeling, RNN, LSTM, transformers, physical learning, catchment, runoff.**

I. Introduction

THE genesis of rainfall-runoff modeling can be traced back to the mid-19th century, originating from a concurrence of growing urbanization and industrialization engineering challenges, such as the construction of urban sewer systems, drainage for reclaimed land, and the design of spillways for reservoirs. These challenges spurred the initial efforts of the field, necessitating a systematic approach to runoff modeling and prediction [1, 2]. Steady developments have occurred since, particularly in the late 20th century with its embrace of advent computational tools and the surge of available data.

The popular and useful conceptual and empirical models of the 1960s and 70s [3–7] have fundamentally contributed to the field [1], and are still widely in use today [8, 9]. However, perhaps more than ever, they are now contending with intrinsic limitations. In the face of more frequent sporadic extreme weather and the intricate hydrological processes it attempts to model, the simplicity that once signified their strength has now become a constraint. Rigid structures and an excess of parameters often lead to a masking of the true hydrological processes with non-physical interpretations and makes the model calibration process non-trivial due to phenomena such as equifinality—non unique solutions. These circumstances compromise the model's reliability and diminishes their utility especially under abnormal climate conditions and evolving land-use patterns.

To solve these flaws several measures have been employed. In broader terms the hydrological community has collectively formulate 23 unresolved problems in the field [10] to unanimate its progress. Among these, Problem 20 explicitly addresses some of the critical issues mentioned above: "How can we disentangle and reduce model structural/parameter/input uncertainty in hydrological prediction?" [10]. This inquiry underscores a pivotal area of research aimed at enhancing the precision and reliability of hydrological models. Concurrently, and in more concrete ways, efforts such as the flexible SHyFT model [11], and the application of machine learning (ML) techniques, notably the use of Long Short-Term Memory (LSTM) [12] networks [13–18], have demonstrated significant potential to address these challenges. In particular, the application of LSTM networks represents a paradigm shift towards exploiting today's great computational power and abundance of data to capitalize on their ability to assume less of the underlying physical phenomena of the process and utilizing their inherent capability to capture complex patterns buried in large datasets [12, 15].

Thus, this paper will explore the use of LSTMs on a Norwegian dataset, seeking to overcome the limitations of traditional hydrological models by leveraging the LSTM's ability to process sequential data and learn from both long and short term dependencies inherent in meteorological patterns. Furthermore, the LSTM's proficiency in modeling complex interactions within the data allows it to learn underlying physical processes such as Snow Water Equivalent (SWE), which are crucial for accurate predictions in snow-dominated regions. By integrating features like SWE, which significantly influence runoff processes, LSTMs can provide a more nuanced understanding of hydrological dynamics, enhancing model performance under varied climatic conditions. This study aims to demonstrate the LSTM's potential not only in capturing temporal patterns but also in effectively incorporating critical hydrological indicators to improve the robustness and reliability of hydrological forecasts.

II. Methods and Data

A. LSTM

A recurrent neural network (RNN) [19] is a network specifically made for sequences or one dimensional structured data such as sentences or time series. The term 'recurrent' stems from the cyclic structure within the RNN architecture. Notably, each hidden node is dependent on the preceding hidden node to pertain past information from the sequence. The hidden nodes are said to "remember" past information. The sequence is introduced in to the network such that each element in the sequence, denoted $\mathbf{x}^{(t)}$, is only connected to the hidden node, denoted $\mathbf{h}^{(t)}$, at the same time step. Here t is the time step (or index of the sequence for other than time series applications). This is in contrast to the "vanilla" neural network, also called Fully Connected Neural Network (FCNN), where all nodes are connected to all other nodes in the subsequent layer. Figure 1 illustrates a schematic depiction of the RNN architecture. To update the hidden nodes, we compute

$$\mathbf{h}^{(t)} = \sigma(\mathbf{W}_{xh}\mathbf{x}^{(t)} + \mathbf{W}_{hh}\mathbf{h}^{(t-1)} + \mathbf{b}),$$

where $\mathbf{h}$ is the hidden node, σ is the sigmoid activation function, W_x and W_h are the weight matrices, and b is the bias. The hidden node is then sent further to compute $\mathbf{h}^{(t+1)}$ and used to compute the output

$$\mathbf{o}_t = \sigma(\mathbf{W}_{ho}\mathbf{h}^{(t)} + \mathbf{b}_o).$$

To update the weights, we have to back-propagate through the network. The back-propagation algorithm for RNNs is called Back-propagation through time. Since

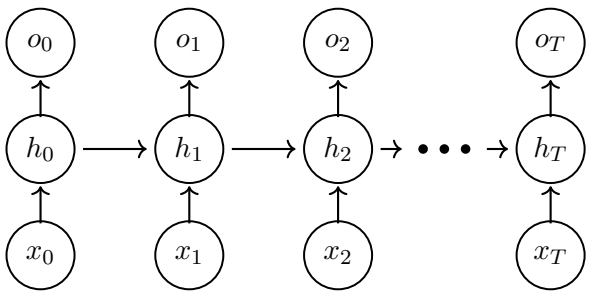

Figure 1: Schematic depiction of RNN architecture with sequence length T

PyTorch solves this automatically, we will not describe it in great detail. The overall loss is the sum of all the loss functions from each time steps.

$$L = \sum_{t=1}^{T} L^{(t)}.$$

Differentiating with respect to the weights and using the chain rule, we end up with the expression

$$\frac{\partial L^{(t)}}{\partial \mathbf{W}_{hh}} = \frac{\partial L^{(t)}}{\partial \mathbf{o}^{(t)}} \cdot \frac{\partial \mathbf{o}^{(t)}}{\partial \mathbf{h}^{(t)}} \cdot \Big(\sum_{k=1}^{t} \frac{\partial \mathbf{h}^{(t)}}{\partial \mathbf{h}^{(k)}} \cdot \frac{\partial \mathbf{h}^{(k)}}{\partial \mathbf{W}_{hh}}\Big),$$

where

$$\frac{\partial \mathbf{h}^{(t)}}{\partial \mathbf{h}^{(k)}} = \prod_{i=k+1}^{t} \frac{\partial \mathbf{h}^{(i)}}{\partial \mathbf{h}^{(i-1)}}.$$

When the model tries to remember too long into the past, we run into the problem of exploding and vanishing gradients. As we can see from the equations above, the expression $\frac{\partial \mathbf{h}^{(t)}}{\partial \mathbf{h}^{(k}}$ has $t-k$ multiplications and thus, if $|w_{hh}| < 1$ it will eventually vanish, and if $|w_{hh}| > 1$ it will eventually explode. This sets a limit to how far back an RNN can "remember" [20].

To deal with this problem, we use an extension of the traditional RNN, called a Long Short-Term Memory introduced (LSTM) network published in 1997 by Sepp Hochreiter and Jürgen Schmidhuber [12], and which has been shown to perform well in hydrological applications [15]. The LSTM model represents a variant of the conventional RNN architecture, retaining a similar overall structure while differing in the computation of the hidden nodes. The LSTM introduces the concept of a cell (or LSTM cell), with a cell state. We can model this cell mathematically as a function. Let

$$g : \mathbb{R}^d \times \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}^p \times \mathbb{R}^p,$$

be a function, where d is the number of features in $\mathbf{x}^{(t)}$ and p the hidden size, then

$$(\mathbf{h}^{(t)}, \mathbf{C}^{(t)}) = g(\mathbf{x}^{(t)}, \mathbf{h}^{(t-1)}, \mathbf{C}^{(t-1)}).$$

The cell state $\mathbf{C}$ represents the long term memory of the network, while the hidden state $\mathbf{h}$ represents the short term memory. The forward computations inside the cell is thus computed as

$$\mathbf{f}_t = \sigma(\mathbf{W}_{xf}\mathbf{x}^{(t)} + \mathbf{W}_{hf}\mathbf{h}^{(t-1)} + \mathbf{b}_f),$$

$$\mathbf{i}_t = \sigma(\mathbf{W}_{xi}\mathbf{x}^{(t)} + \mathbf{W}_{hi}\mathbf{h}^{(t-1)} + \mathbf{b}_i),$$

$$\hat{\mathbf{C}}_t = \tanh(\mathbf{W}_{xc}\mathbf{x}^{(t)} + \mathbf{W}_{hc}\mathbf{h}^{(t-1)} + \mathbf{b}_c),$$

$$\mathbf{C}^{(t)} = (\mathbf{C}^{(t-1)} \odot \mathbf{f}_t) \oplus (\mathbf{i}_t \odot \hat{\mathbf{C}}_t),$$

$$\mathbf{o}_t = \sigma(\mathbf{W}_{xo}\mathbf{x}^{(t)} + \mathbf{W}_{ho}\mathbf{h}^{(t-1)} + \mathbf{b}_o),$$

$$\mathbf{h}^{(t)} = \mathbf{o}_t \odot \tanh(\mathbf{C}^{(t)}),$$

where σ and $\tanh$ are activation functions, $\odot$ is the element-wise multiplication operator and $\oplus$ is the element-wise addition operator. The matrices $\mathbf{f}_t$, $\mathbf{i}_t$, $\hat{\mathbf{C}}_t$ and $\mathbf{o}_t$ are what we call gates. The LSTM model employs these gates within its architecture to effectively regulate the information flow. Each gate has a specific role, controlling whether to retain or forget certain information at each time step. These gates are critical for the LSTM's ability to handle long-term dependencies by maintaining a balance between input data, memorizing relevant past information, and predicting future events. Below is a detailed explanation of each of the gates involved in the LSTM computations:

1. Forget Gate ($\mathbf{f}_t$)

The forget gate decides which information should be discarded from the cell state. This decision is made using the sigmoid activation function (σ), which outputs values between 0 and 1. These values are multiplied element-wise ($\odot$) with the previous cell state ($\mathbf{C}^{(t-1)}$):

- **0** indicates that the information should be completely forgotten.
- **1** indicates that the information should be entirely retained.

Mathematically, it is defined as:

$$\mathbf{f}_t = \sigma(\mathbf{W}_{xf}\mathbf{x}^{(t)} + \mathbf{W}_{hf}\mathbf{h}^{(t-1)} + \mathbf{b}_f),$$

where $\mathbf{W}_{xf}$ and $\mathbf{W}_{hf}$ are the weight matrices for the input and previous hidden state, respectively, and $\mathbf{b}_f$ is the bias.

2. Input Gate ($\mathbf{i}_t$)

The input gate decides which new information will be stored in the cell state. It works in conjunction with the candidate memory ($\hat{\mathbf{C}}_t$), which proposes a value to add to the cell state. The input gate's output, also determined by a sigmoid function, specifies the update's importance:

- **0** indicates no update to the cell state at this index.
- **1** indicates a full update at this index.

The candidate memory ($\hat{\mathbf{C}}_t$), proposed new values for the cell state, is computed using the hyperbolic tangent function ($\tanh$), which outputs values between -1 and 1, suggesting how much to increase or decrease the cell state at each element:

$$\mathbf{i}_t = \sigma(\mathbf{W}_{xi}\mathbf{x}^{(t)} + \mathbf{W}_{hi}\mathbf{h}^{(t-1)} + \mathbf{b}_i),$$

$$\hat{\mathbf{C}}_t = \tanh(\mathbf{W}_{xc}\mathbf{x}^{(t)} + \mathbf{W}_{hc}\mathbf{h}^{(t-1)} + \mathbf{b}_c),$$

3. Output Gate ($\mathbf{o}_t$)

The output gate controls what the next hidden state ($\mathbf{h}^{(t)}$) should be, which is used in the output of the LSTM unit and for predicting future states. This gate filters the parts of the cell state to output through another sigmoid function. The final hidden state for this time step is then computed by multiplying ($\odot$) this gate's output with the $\tanh$ of the cell state, providing the next output and hidden state of the LSTM:

$$\mathbf{o}_t = \sigma(\mathbf{W}_{xo}\mathbf{x}^{(t)} + \mathbf{W}_{ho}\mathbf{h}^{(t-1)} + \mathbf{b}_o),$$

$$\mathbf{h}^{(t)} = \mathbf{o}_t \odot \tanh(\mathbf{C}^{(t)}),$$

Final Cell State Update

The cell state at each time step ($\mathbf{C}^{(t)}$) is updated by merging the previous cell state with new candidate values, controlled by the forget and input gates:

$$\mathbf{C}^{(t)} = (\mathbf{C}^{(t-1)} \odot \mathbf{f}_t) \oplus (\mathbf{i}_t \odot \hat{\mathbf{C}}_t),$$

where $\oplus$ represents element-wise addition.

Thus, these gates enable the LSTM to selectively remember and forget information, which is crucial for tasks involving sequences with long-range dependencies, like hydrological modeling or language modeling. By learning which data is important to keep or discard, LSTMs can make more accurate predictions and adapt to the intricacies of specific data patterns.

In our specific application, the LSTM operates as a many-to-one model, where sequences of input data are processed to produce a single output that corresponds to the target value at the final time step. To adapt the LSTM output to match the target format, the final output from the LSTM, $\mathbf{h}^{(T)}$, where T is the final time step, is passed through a dense feed-forward neural network (FFNN). This dense layer transforms the LSTM's high-dimensional output, of [sequence_length,

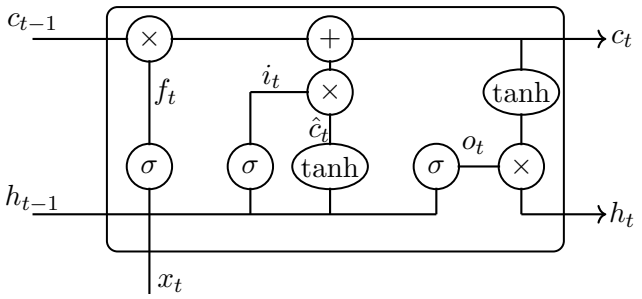

Figure 2: Schematic of an LSTM cell architecture illustrating the flow and transformation of information through its various components. The cell state (c_{t-1}) and hidden state (h_{t-1}) from the previous time step pass through different gates—forget gate (f_t), input gate (i_t), and output gate (o_t)—which regulate the retention, updating, and output of the memory state. The gates utilize sigmoid (σ) and $\tanh$ activation functions to modulate the information flow, leading to the updated cell state (c_t) and output hidden state (h_t).

batch_size, hidden_size], to fit the target's dimensions and scale, a scalar value of the daily discharge in our case. This typically consists of a linear transformation followed by a non-linear activation function, allowing the model to fine-tune its predictions to closely align with expected outcomes. This structure leverages the LSTM's capability to capture relevant temporal features with the dense layer's specific output adaptation, optimizing the overall prediction accuracy.

Figure 2 illustrates a schematic depiction of the internal LSTM architecture and how its gates interact to mold the input data, $\mathbf{x}^{(t)}$, previous hidden state, $\mathbf{h}^{(t-1)}$, and previous cell state, $\mathbf{c}^{(t-1)}$, to produce the next hidden and cell states, $\mathbf{h}^{(t)}$ and $\mathbf{c}^{(t)}$ respectively.

B. Study Area and Data

This study focuses on Catchment 088, Strynvassdraget/Indre Nordfjord, located in the mountainous region of Western Norway, adjacent to the Jostedal Glacier, the largest glacier in continental Europe. See Figure 3 for the catchment placement in Norway. The catchment features significant geographic diversity, including deep fjords and substantial altitude gradients. It encompasses a land area of 1123 km^2 and a total area of 1151 km^2, with altitudes ranging from 0 m to a maximum of 1935 m above sea level and with an altitude mean of 982 m. This region was selected due to its lowest regulation status among all catchments in southern Norwegian, i.e. very low to no human tampering with the hydrological aspects of the catchments, e.g. reservoirs.

The hydrological data was provided by the Norwegian Water Resources and Energy Directorate's (NVE) HydAPI [21] and combines daily discharge from three stations into a single dataset representing the catchment's foot, as illustrated in Figure 4.

High-resolution wind data was sourced from the Norwegian Meteorological Institute's KliNoGrid dataset [22]. Snow Water Equivalent (SWE) data, derived from the seNorge dataset [23], quantify the water content of snowpacks. This measurement is essential for estimating snowmelt contributions to the hydrological cycle, particularly in mountainous terrains where such contributions are significant. Daily minimum and maximum temperatures and precipitation volumes were also retrieved from the seNorge archive [24]. These parameters are fundamental for calibrating hydrological models against observed climatological trends. Additionally, relative humidity, shortwave radiation, and surface pressure data were acquired from a dataset hosted on Zenodo [25].

All meteorological features are processed to yield daily and annual averages across the catchment. This aggregation facilitates a coherent integration of climatological impacts on hydrological phenomena, thereby enhancing the accuracy and reliability of the LSTM-based hydrological model outlined in subsection II-A.

To systematically examine the impact of training data length and feature set on the model's performance, six datasets were created: three datasets contain all features with varying training data lengths—full dataset spanning the entire training period, one covering three-quarters, and another spanning half the duration. Additionally, three corresponding datasets were prepared by entirely removing the SWE feature to assess its impact. The training set spans from 15 March 1987 to 30 December 2000, and the remaining data from 1 January 2007 to 30 May 2015 is evenly divided between validation and testing, with the test set being slightly larger.

C. Evaluation Metrics

This section outlines the evaluation metrics employed for assessing the performance of the hydrological model. We primarily utilize the Mean Squared Error (MSE) and the Nash-Sutcliffe Efficiency (NSE), which are standard benchmarks in hydrological modeling due to their effectiveness in quantifying the accuracy of simulated versus observed data [26].

1) Mean Squared Error (MSE)

The MSE is commonly used to measure the average squared difference between the observed and predicted values in machine learning, and it is the metric used as the loss function during the training of our models. It is defined as:

$$\text{MSE} = \frac{1}{n}\sum_{t=1}^{n}(x_{s,t} - x_{o,t})^2, \quad (1)$$

where $x_{s,t}$ denotes the simulated value at time t, and $x_{o,t}$ is the corresponding observed value. Lower MSE values indicate better model performance, with a value of 0 representing a perfect fit.

2) Nash-Sutcliffe Efficiency (NSE)

The NSE is a normalized statistic used to assess the predictive power of hydrological models. It compares the residual variance to the variance of the observed data:

$$\text{NSE} = 1 - \frac{\sum_{t=1}^{n}(x_{s,t} - x_{o,t})^2}{\sum_{t=1}^{n}(x_{o,t} - \mu_o)^2} = 1 - \frac{\text{MSE}}{\sigma_o^2}, \quad (2)$$

where μ_o and σ_o^2 represent the mean and variance of the observed data, respectively. The NSE is very similar to the R-squared (coefficient of determination) metric, but does not have the same statistical correlation between $x_{s,t}$ and $x_{o,t}$ as R-squared. Non the less, an NSE value of 1 indicates perfect predictions, 0 means the model predictions are as accurate as the average of the observed data, and negative values suggest poorer performance.

3) Interpretation of Metrics

The metrics described provide a comprehensive evaluation of model performance. MSE focuses on the magnitude of errors and serves as the basis for the loss function during model training. In contrast, NSE provides a dimensionless evaluation that helps compare performance across different datasets and contexts. Together, they offer valuable insights into different facets of model accuracy and guide further refinements in hydrological modeling.

4) Visual Evaluation of Hydrographs

In addition to quantitative metrics, visual evaluation of hydrographs is employed to assess the temporal dynamics and peak flow accuracies of the models. Hydrographs are particularly valuable for identifying model bias and differences in the timing and magnitude of peak flows, as well as the shape of recession curves [27]. This qualitative assessment complements the quantitative metrics by providing an intuitive evaluation

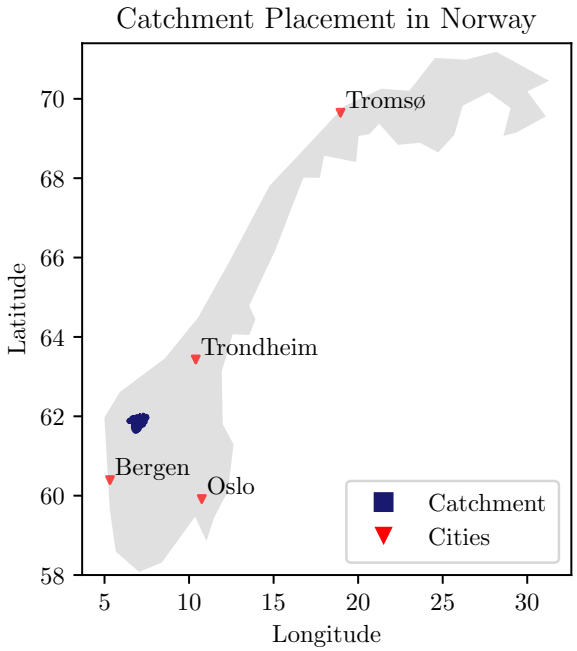

Figure 3: **Catchment Placement in Norway.** This figure illustrates the catchment's location (in blue), and some major cities (red triangles), across Norway.

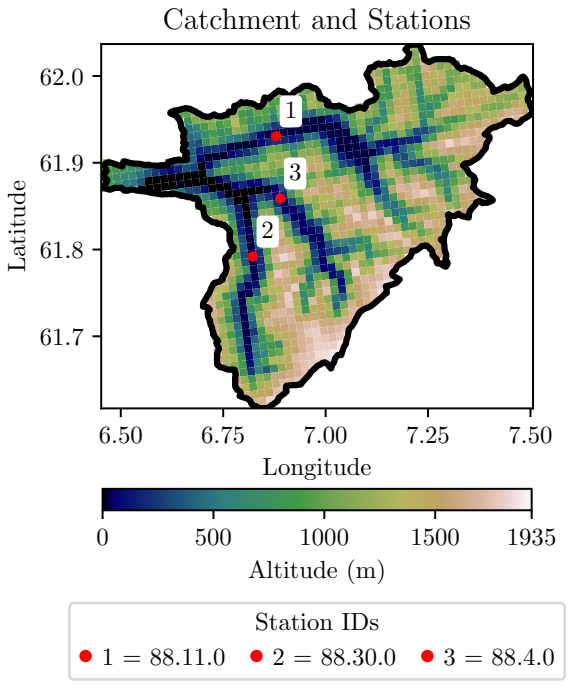

Figure 4: **Catchment and Stations.** This figure displays the distribution of hydrological stations within the catchment area, overlaid on a altitude gradient map.

of the model's ability to capture essential hydrological behaviors and variations over time. Moriasi et al. [28] highlights the importance of graphical techniques in model evaluation, advocating for their use alongside traditional statistical metrics.

5) Correlation Analysis

Additionally, the correlation between LSTM cell states and input data features, particularly meteorological variables, was a crucial aspect of model evaluation. We utilized the `numpy.corrcoef` function from the NumPy library to compute the Pearson correlation coefficients. The Pearson correlation coefficient is calculated as:

$$r = \frac{\sum_{i=1}^{n}(X_i - \overline{X})(Y_i - \overline{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \overline{X})^2 \sum_{i=1}^{n}(Y_i - \overline{Y})^2}} \tag{3}$$

where X and Y are the variables being compared, and $\overline{X}$ and $\overline{Y}$ are the means of X and Y respectively. This coefficient provides a statistical measure of the strength and direction of the linear relationship between the cell states and the corresponding input features [29]. A correlation coefficient close to 1 indicates a strong positive relationship, close to -1 indicates a strong negative relationship, and around 0 indicates no linear relationship. High correlation values suggest that the LSTM effectively encodes relevant information in its cell states, essential for accurate hydrological forecasting. Conversely, low or near-zero correlations might indicate that important dynamics are not being captured by the model, necessitating further analysis and potential model adjustments. Such correlations also help in identifying which features most significantly influence the model's output, guiding further model refinement and feature selection. This analytical approach contributes to a deeper understanding of the model's internal representations and its ability to learn complex dependencies within the training data.

D. Tuning and Calibration

To address the complexities involved in hydrological forecasting with LSTM networks, this study employs Ray Tune [30], a state-of-the-art hyperparameter tuning framework. Ray Tune provides a scalable and efficient mechanism for searching through hyperparameter spaces, leveraging advanced optimization algorithms and distributed computing to expedite the tuning process. The choice of Ray Tune is motivated by its ability to automate the trial-and-error process of model calibration, thereby reducing the computational time and resources required to identify optimal model configurations.

Moreover, the tuning process, aimed to explore a broad spectrum of model architectures and parameters, involved the exploration of the following hyperparameter:

- **Hidden Size:** Selection from {128, 192, 256, 320}, impacting the model's capacity to capture complex dependencies.
- **Number of Layers:** Only 1 and 2 layers were tested as model complexity grows quickly with this parameter in LSTMs, and overfitting becomes an issue.
- **Dropout Rate:** Varied among {0.1, 0.2, 0.3, 0.4, 0.5}, to prevent overfitting by 1) randomly excluding unit connections between the LSTM layers if two layers are used, and 2) by randomly shutting down nodes in the FFNN during training.
- **Learning Rate:** Searched over a log-uniform distribution between 10^{-4} and 10^{-2}.
- **Batch Size:** Examined across {128, 256, 512, 1024, 2048}, affecting the stability and speed of the learning process.
- **Lookback Period or sequence length:** Chosen from {240, 270, 300, 365, 400} days, determining the window of historical data for making predictions.
- **L1 regularization:** The λ_1 scaling coefficient for the L1 regularization was samples from 10^{-9} to 10^{1} with jumps of 10^{1}.
- **L2 regularization:** Similarly the scaling coefficient λ_2 for the L2 regularization was also samples between 10^{-9} to 10^{1} with jumps of 10^{1}.

Ray Tune integrates with the model training loop to systematically evaluate the performance of different hyperparameter configurations against the loss of the validation data set. Through its efficient scheduling and early stopping algorithms, such as ASHA (Asynchronous Successive Halving Algorithm), Ray Tune effectively narrows down the search space to the most promising configurations, thereby optimising computational efficiency.

The hyperparameter search for our LSTM setup did not yield one unique combination that triumphed the others. On the contrary, a somewhat broad space of about three distinct values per hyperparameter gave similar results. Thus, with seven parameters, excluding the number of layers, we have $3^7 = 2187$ different combinations that all lead to similar results. And, with training time of about 0.5 to 1.5 second per epoch and with early stopping on the validations loss kicking in at around 150 - 250 epochs we get 200 seconds $\times\, 2187$ runs $= 437400$ seconds, or about

122 hours worth of exploring. Thus, the last part of the tuning was done by hand and with intuition.

Therefore, a hidden size of 256 gave good results and was also used by Kratzert [17], the number of layers that yielded the best results where 2, the dropout rate 20% and the learning rate was set to 0.007. Even though a batch size of 512 gave the lowest MSE and highest NSE, the batch size that gave the smoothest loss and most consistent results was 2048, and was therefore used. The high batch size was also the fastest. The lookback period was intuitively set to 365, giving the model a one year memory. Regarding the L1 and L2 regularization, a combination of the two was used. The L1 term was set to 0.001 a bit lower than the L2 term which was set to 0.1. A reversed combination of 0.1 and 0.001 was also proficient but the former was chosen to preserve as many features as possible for later interpretability.

E. Experimental Design

The experimental design of this study was structured to systematically evaluate the performance of an LSTM-based model under various configurations of data availability and feature inclusion, to discern the impact of these variables on the model's accuracy and learning capabilities. This approach entailed conducting a series of six trials, each designed to assess different aspects of model behavior and robustness. The trials ranged from using the complete dataset with all features across the entire training period, serving as a baseline, to progressively reducing the data volume and feature set. Specifically, the trials included using all available features over the entire training period, half of the training period, and a quarter of the training period, each scenario both with and without the snow water equivalent (SWE) feature.

The architectural parameters and hyperparameters were kept consistent across all trials to isolate the effects of data variations. The chosen architecture comprised a hidden size of 256, a lookback period of 365 days, two layers, and a batch size of 2048. The learning rate was set at 0.007, with dropout between layers maintained at 0.2 and L1 and L2 regularization scaled at 0.01 and 0.1, respectively. These settings were determined using the full dataset encompassing all features, which helped in standardizing the evaluation metrics across different datasets.

The primary goals of these experiments were to evaluate the model's ability to predict daily discharge accurately across varying data volumes and to understand the influence of specific meteorological features on the model's predictive power. This included assessing how the exclusion of SWE affected the model's discharge predictions and its capability to infer underlying hydrological processes. Additionally, the model's capacity to learn and represent key meteorological features such as precipitation, temperature, shortwave radiation, and SWE in its cell states was scrutinized, particularly focusing on the correlation of cell states with SWE under different data conditions. This analysis aimed to reveal how reductions in data availability impact the model's ability to capture and mimic the behavior of crucial hydrological features.

Inspired by Kratzert et al. (2018), who pioneered the extraction and analysis of LSTM cell states to interpret the learning mechanisms within rainfall-runoff modeling, this study extends the methodology to explore the LSTM's ability to implicitly learn and represent physical processes, such as changes in SWE over time, even in the absence of explicit data on these features. This aspect of the study was particularly focused on examining the model's robustness and adaptability under constrained data environments, providing valuable insights into the potential and limitations of deep learning models in hydrological forecasting.

F. Open-source Software and Hardware

The computational experiments presented in this study are implemented using a collection of open-source software, ensuring reproducibility and accessibility of our methods. The core programming environment is based on Python 3.10 [31]. Data preprocessing and management are performed with Numpy [32], Pandas [33] and GeoPandas [34], while Scikit-Learn [35] supports additional data analysis tasks. PyTorch [36] is employed for the development and training of the LSTM networks, and Ray Tune [30] is used for model architecture and hyperparameter optimization. Visualizations are created using Matplotlib [37].

The computational experiments were conducted on the "ml9" node within the AI cluster at the University of Oslo. This node is part of the UiO AI Hub's dedicated resources for machine learning and deep learning research [38]. Equipped with four NVIDIA GeForce RTX 3090 GPUs, each with 24 GB of memory, the "ml9" node provides substantial computational power and memory capacity. These features are crucial for efficiently handling large datasets and complex computations required in

training deep neural networks. The utilization of such advanced hardware, with its high power capacity and CUDA Version 12.3 support, ensures rapid processing and enhances our research's capability to conduct large-scale, intensive model training and data analysis sessions. The strategic use of these high-performance computing resources significantly contributes to minimizing computational time and maximizing the productivity and scalability of our hydrological modeling efforts.

III. Results

The results of this study showcase the robustness and adaptability of the LSTM-based model in hydrological forecasting under various data configurations and feature set modifications. Each experiment was designed to assess the model's ability to predict daily discharge and learn the behavior of snow water equivalent (SWE) in the catchment—a critical meteorological feature in modeling snow-covered catchments. The impact of excluding SWE from the training data was also evaluated.

A. Model Performance on Daily Discharge Prediction

The LSTM model consistently demonstrated robust performance, as evidenced in Figure 5. The highest accuracy was achieved when utilizing half of the dataset, achieving an NSE of 0.90, closely followed by the full dataset with an NSE of 0.89. The model trained on three-quarters of the dataset showed a modest decrease in performance but still maintained a commendable NSE of 0.87. According to [28], an NSE score greater than 0.5 is considered satisfactory. While all models closely mirrored the observed discharge patterns, they encountered difficulties in accurately predicting the most volatile peaks in the hydrograph.

B. Learning Meteorological Features and Impact of Excluding SWE

Analysis of the LSTM cell states revealed their significant capacity to learn and accurately reflect key meteorological features. For instance, correlations for min/max temperature ranged from 0.83 to 0.89 in models trained with SWE, and from 0.81 to 0.83 in those without SWE. Similarly, correlations with humidity were between 0.83 to 0.85, and 0.80 to 0.81, and for shortwave radiation, 0.76 to 0.80 and 0.64 to 0.65 respectively. These results indicate minor discrepancies in the models' ability to model and represent the input features, regardless of SWE inclusion.

Furthermore, models trained on the full and three-quarters datasets exhibited strong correlations with the SWE feature, both registering 0.80. However, a decline in correlation to 0.67 was observed for the model trained

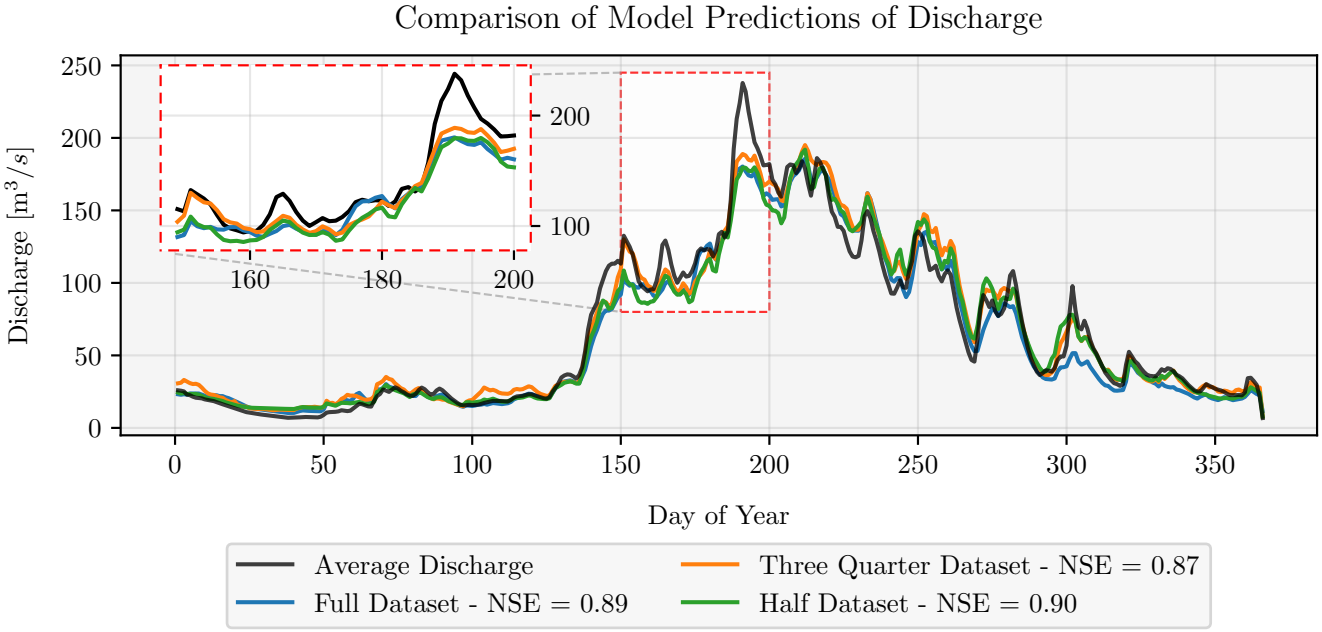

Figure 5: **Comparison of model predictions of discharge using different training periods.** The black line represents the actual data yearly average. Models trained on full yearly average (NSE = 0.89), three-quarter yearly average (NSE = 0.87), and half-yearly average (NSE = 0.90) are shown in blue, orange, and green, respectively.

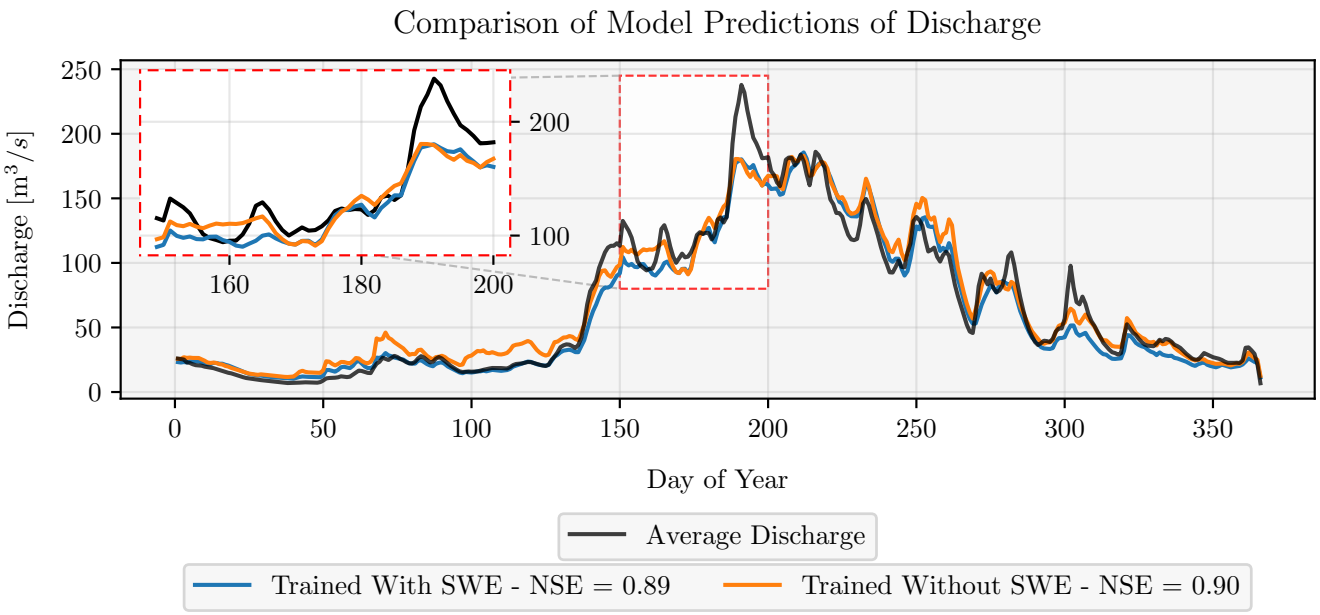

Figure 6: **Comparison of model predictions of discharge with and without snow water equivalent (SWE) data.** The black line represents the actual data yearly average. Models trained with SWE (NSE = 0.89) and without SWE (NSE = 0.90) are shown in blue and orange, respectively.

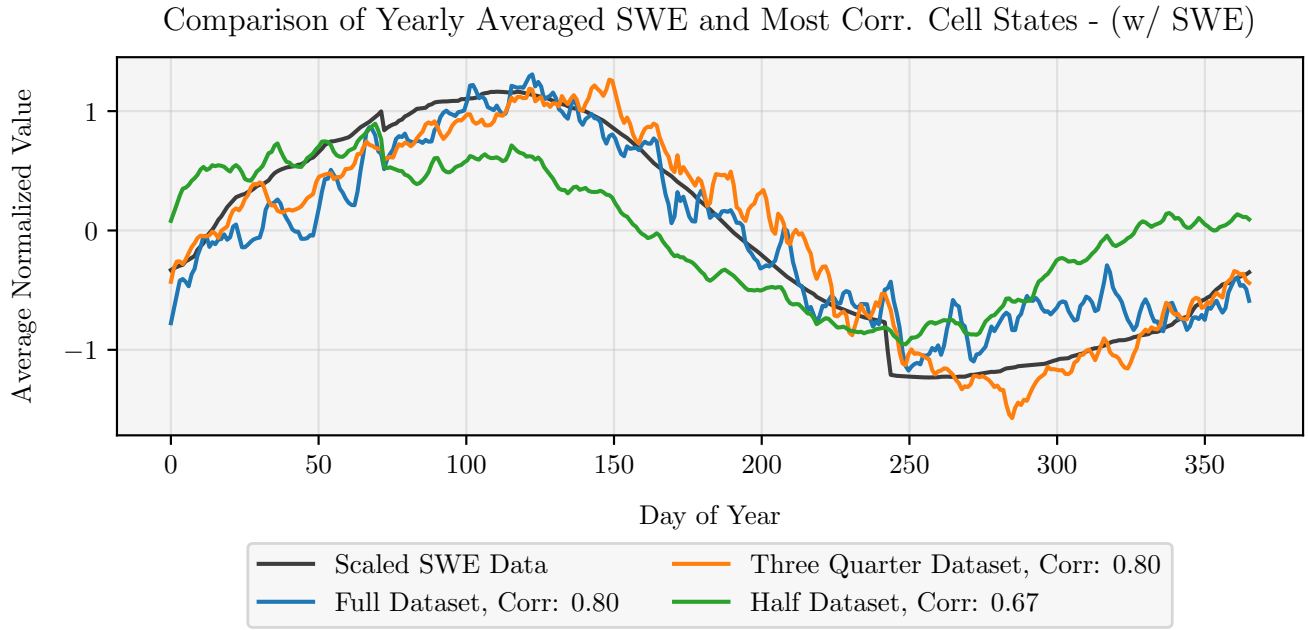

Figure 7: **Comparison of yearly averaged SWE and the most correlated cell states for models trained with SWE.** The black line represents the normalized feature series, while the most correlated cell states for full (NSE = 0.80), three-quarter (NSE = 0.80), and half (NSE = 0.67) dataset models are shown in blue, orange, and green, respectively.

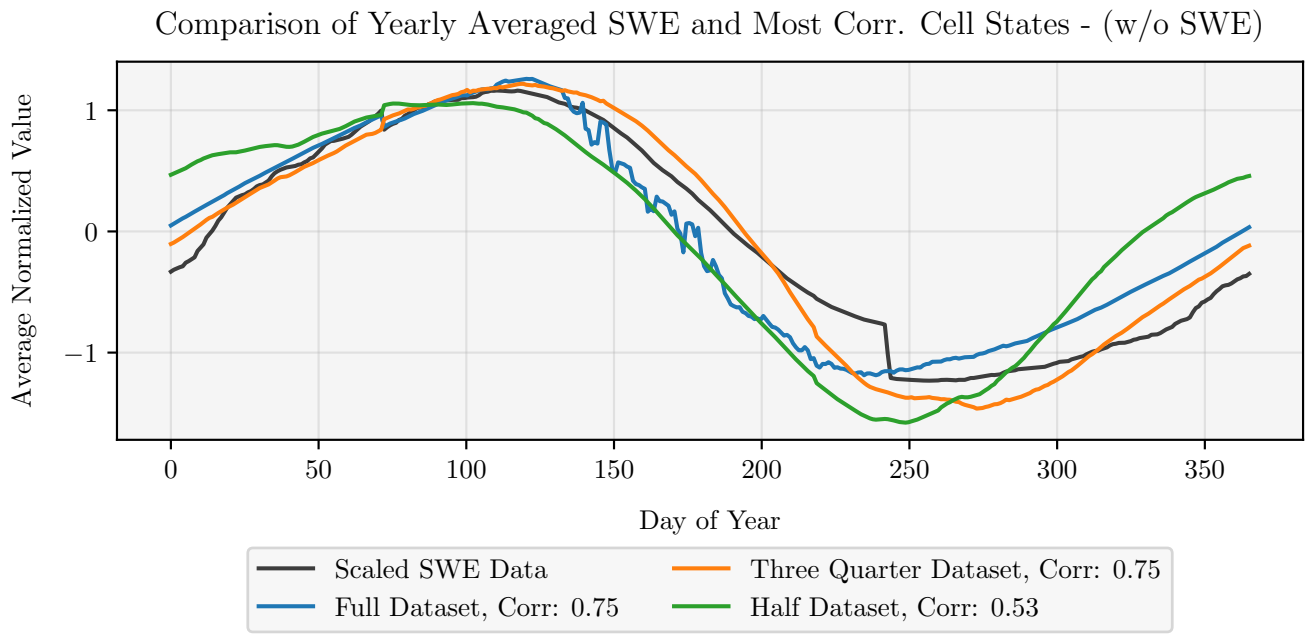

Figure 8: **Comparison of yearly averaged SWE and the most correlated cell states for models trained without SWE.** The black line represents the normalized feature series, while the most correlated cell states for full (NSE = 0.75), three-quarter (NSE = 0.75), and half (NSE = 0.53) dataset models are shown in blue, orange, and green, respectively.

with only half of the dataset. A similar pattern emerged in models trained without SWE, showing correlations of 0.75 for both full and three-quarters datasets, and a lower correlation of 0.53 for the half dataset model.

Notably, the cell states in models trained with SWE exhibited greater volatility compared to those trained without, which presented smoother states.

The exclusion of Snow Water Equivalent (SWE) from the training dataset had an unexpected impact on model performance. Contrary to initial assumptions that the inclusion of SWE would enhance model accuracy due to its hydrological significance, the models trained without this feature actually demonstrated superior performance. Specifically, the Nash-Sutcliffe Efficiency (NSE) for the model excluding SWE reached 0.90, surpassing the 0.89 NSE observed in the model that included SWE. This suggests that the LSTM was able to effectively compensate for the absence of SWE data by possibly relying on other correlated features within the dataset or by benefiting from reduced model complexity and overfitting risks.

IV. Discussion

A. Model Efficiency and Data Dependency

The robust performance across LSTM models trained with varying data volumes underscores the architecture's intrinsic ability to capture and represent essential hydrological dynamics effectively. This study confirms that the minimal performance variation between models trained with and without Snow Water Equivalent (SWE) is indicative of the LSTM's capability to mimic significant physical processes of the hydrological cycle, even without direct exposure to all relevant features. This characteristic enhances the model's utility in scenarios with incomplete data sets, where traditional models may falter.

B. Significance of Data Volume

The analysis revealed that while larger datasets generally improve model accuracy, the LSTM's performance with reduced data volumes was remarkably high. Models trained with only half the dataset still achieved near-optimal performance, challenging the traditional assumption that more data invariably leads to better performance. This suggests a fundamental efficiency of the LSTM in generalizing from limited data, which is particularly advantageous for applications in regions where comprehensive data collection is challenging or infeasible.

C. Implicit Learning of SWE Dynamics

The ability of the LSTM to implicitly learn and represent the dynamics of SWE, even when it was not included in the dataset, was an impressive outcome of this study. This capability was more pronounced in models trained with more extensive datasets. As the amount of training data decreased, the model's effectiveness in mimicking the SWE dynamics gradually diminished, affirming the importance of data volume in capturing the dynamics of the hydrological cycle effectively. These findings underscore the critical relationship between dataset size and the depth of learned processes, which is crucial for the accurate simulation of runoff processes.

D. Implications for Hydrological Forecasting

Our findings advocate for the adoption of LSTM models in hydrological forecasting, especially in catchments with sparse data. The ability of LSTMs to maintain high accuracy despite reduced data volumes and the absence of critical features like SWE offers a promising avenue for deploying these models in varied hydrological scenarios. Additionally, the effective simulation of snow behavior and the strong correlation of LSTM cell states with SWE, particularly in full and three-quarters data-trained models, further corroborate the robustness of the LSTM approach.

E. Learning Meteorological Features and Impact of Excluding SWE

The LSTM models exhibited strong capabilities in capturing and learning from the meteorological features essential for hydrological modeling. Notably, even when SWE was excluded from the training dataset, the models adeptly predicted discharge and other hydrological responses, underscoring the adaptability and resilience of this modeling approach. The analysis of LSTM cell states showed that they maintained significant correlations with key meteorological variables under different data conditions, highlighting the model's ability to infer important hydrological signals from available data.

F. Overall Assessment

The comprehensive analysis conducted through various experimental setups demonstrated the LSTM's capability to handle both complete and constrained datasets without substantial loss in performance. These results are particularly valuable for regions facing logistical or financial constraints in data collection, suggesting that LSTMs can significantly enhance forecasting accuracy with even limited data inputs.

V. Conclusion

This study has demonstrated the robustness and versatility of LSTM networks in modeling hydrological processes, particularly in environments where traditional models might struggle due to limited data availability or complex dynamic processes that are difficult to capture with standard techniques. The LSTM's ability to infer significant hydrological behaviors from limited datasets, and its performance under various configurations of data availability and feature inclusion, marks a substantial advancement in the field of hydrological modeling.

The findings underscore the potential of LSTM models to perform reliably across different hydrological scenarios, even when critical data such as Snow Water Equivalent (SWE) are absent. This capability is crucial for regions where data collection faces logistical and financial constraints, thereby broadening the scope for applying advanced machine learning techniques in hydrological forecasting.

Moreover, the study highlighted the importance of dataset size in enhancing the model's ability to learn and represent complex hydrological processes accurately. While larger datasets generally improve model performance, the LSTM's capacity to maintain high accuracy with reduced datasets is particularly noteworthy. This suggests that LSTM models can be effectively used in less monitored or data-scarce regions, providing reliable forecasts and aiding in water resource management and planning.

Future work should focus on further exploring the limits of data reduction in training robust models and expanding the application of LSTM models to other aspects of hydrological modeling such as flood prediction and drought management. Additionally, integrating LSTM models with real-time data streams could pave the way for more dynamic and responsive hydrological forecasting systems.

Overall, the application of LSTM networks in this study reflects a significant step forward in leveraging machine learning for environmental and hydrological studies, promising to enhance our understanding and management of water resources in an increasingly unpredictable global climate.

ACKNOWLEDGMENT

The authors would like to thank Fabio Zeiser and Felix Matt from Statkraft for their generous and enriching guidence. We would also like to thank Kristen Joy Valseth for providing data, scripts for retrieving data and software recommendations.

REFERENCES

[1] C.-Y. Xu, *Hydrological Models*. Oslo, Norway: University of Oslo, 2012, compendium for hydrological model identification.

[2] T. J. Mulvaney, "On the use of self-registering rain and flood gauges in making observations of the relations of rainfall and flood discharges in a given catchment," *Proceedings of the institution of Civil Engineers of Ireland*, vol. 4, no. 2, pp. 18–33, 1851.

[3] D. R. Dawdy and T. O'Donnell, "Mathematical models of catchment behaviour," *Journal of the Hydraulics Division, Proceedings of the American Society of Civil Engineers*, vol. 91, no. HY4, pp. 123–137, 1965.

[4] N. H. Crawford and R. K. Linsley, "Digital simulation in hydrology: Stanford watershed model iv," Stanford University, Department of Civil Engineering, California, Tech. Rep. Technical Report No. 39, 1966.

[5] R. J. C. Burnash, R. L. Ferral, and R. A. McGuire, "A generalised streamflow simulation system – conceptual modelling for digital computers," US Department of Commerce, National Weather Service and State of California, Department of Water Resources, 1973.

[6] S. Bergström and A. Forsman, "Development of a conceptual deterministic rainfall-runoff mode," *Nord. Hydrol*, vol. 4, pp. 240–253, 1973.

[7] W. M. Organization, "Intercomparison of conceptual models used in operational hydrological forecasting," World Meteorological Organization, Tech. Rep. Operational Hydrology Report No. 7, WMO-No. 429, 1975.

[8] J. Seibert and M. J. P. Vis, "A retrospective on hydrological catchment modelling based on half a century with the hbv model," *Hydrology and Earth System Sciences*, vol. 26, pp. 1371–1391, 2022. [Online]. Available: https://hess.copernicus.org/articles/26/1371/2022/

[9] R. J. C. Burnash, "Sacramento model," OpenGMS, 2019. [Online]. Available: https://geomodeling.njnu.edu.cn/modelItem/1b8ccc67-5031-4064-ab1e-712924be8e9c

[10] G. Blöschl, M. F. Bierkens, A. Chambel, C. Cudennec, G. Destouni, A. Fiori, J. W. Kirchner, J. J. McDonnell, H. H. Savenije, M. Sivapalan *et al.*, "Twenty-three unsolved problems in hydrology (uph)–a community perspective," *Hydrological sciences journal*, vol. 64, no. 10, pp. 1141–1158, 2019.

[11] J. F. Burkhart, F. N. Matt, S. Helset, Y. Sultan Abdella, O. Skavhaug, and O. Silantyeva, "Shyft v4. 8: a framework for uncertainty assessment and distributed hydrologic modeling for operational hydrology," *Geoscientific Model Development*, vol. 14, no. 2, pp. 821–842, 2021.

[12] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[13] M. Gauch, F. Kratzert, D. Klotz, G. Nearing, J. Lin, and S. Hochreiter, "Rainfall–runoff prediction at multiple timescales with a single long short-term memory network," *Hydrology and Earth System Sciences*, vol. 25, no. 4, pp. 2045–2062, 2021.

[14] D. Klotz, F. Kratzert, M. Gauch, A. Sampson, J. Brandstetter, G. Klambauer, S. Hochreiter, and G. Nearing, "Uncertainty estimation with deep learning for rainfall–runoff modelling," *Hydrology and Earth System Sciences Discussions*, vol. 2021, pp. 1–32, 2021.

[15] F. Kratzert, D. Klotz, C. Brenner, K. Schulz, and M. Herrnegger, "Rainfall–runoff modelling using long short-term memory (lstm) networks," *Hydrology and Earth System Sciences*, vol. 22, no. 11, pp. 6005–6022, 2018.

[16] F. Kratzert, D. Klotz, M. Herrnegger, A. Sampson, S. Hochreiter, and G. Nearing, "Toward improved predictions in ungauged basins: Exploiting the power of machine learning," *Water Resources Research*, vol. 55, no. 12, pp. 11 344–11 354, 2019.

[17] F. Kratzert, D. Klotz, G. Shalev, G. Klambauer, S. Hochreiter, and G. Nearing, "Towards learning universal, regional, and local hydrological behaviors via machine learning applied to large-sample datasets," *Hydrology and Earth System Sciences*, vol. 23, no. 12, pp. 5089–5110, 2019.

[18] T. Lees, M. Buechel, B. Anderson, L. Slater, S. Reece, G. Coxon, and S. J. Dadson, "Benchmarking data-driven rainfall-runoff models in great britain: A comparison of lstm-based models with four lumped conceptual models," *Hydrology and Earth System Sciences Discussions*, vol. 2021, pp. 1–41, 2021.

[19] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors." *Nature*, vol. 323, no. 6088, pp. 533–536,

1986.

[20] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166, 1994.

[21] "Hydapi documentation," https://hydapi.nve.no/UserDocumentation/, accessed: [insert date of access here].

[22] "Klinogrid dataset," https://thredds.met.no/thredds/catalog/metusers/klinogrid/KliNoGrid_16.12/FFMRR-Nor/catalog.html, accessed: date.

[23] "senorge snow water equivalent data," https://thredds.met.no/thredds/catalog/senorge/seNorge_snow/swe/catalog.html, accessed: date.

[24] "senorge 2018 archive," https://thredds.met.no/thredds/catalog/senorge/seNorge_2018/Archive/catalog.html, accessed: date.

[25] "Zenodo meteorological dataset," https://zenodo.org/records/1970170#.Y9MLlqfMJH4, accessed: date.

[26] H. V. Gupta, H. Kling, K. K. Yilmaz, and G. F. Martinez, "Decomposition of the mean squared error and nse performance criteria: Implications for improving hydrological modelling," *Journal of hydrology*, vol. 377, no. 1-2, pp. 80–91, 2009.

[27] I. ASCE Task Committee on Definition of Criteria for Evaluation of Watershed Models of the Watershed Management Committee and D. Division, "Criteria for evaluation of watershed models," *Journal of Irrigation and Drainage Engineering*, vol. 119, no. 3, pp. 429–442, 1993.

[28] D. N. Moriasi, J. G. Arnold, M. W. Van Liew, R. L. Bingner, R. D. Harmel, and T. L. Veith, "Model evaluation guidelines for systematic quantification of accuracy in watershed simulations," *Transactions of the ASABE*, vol. 50, no. 3, pp. 885–900, 2007.

[29] "numpy.corrcoef," https://numpy.org/doc/stable/reference/generated/numpy.corrcoef.html, accessed: date-of-access.

[30] R. Liaw, E. Liang, R. Nishihara, P. Moritz, J. E. Gonzalez, and I. Stoica, "Tune: A research platform for distributed model selection and training," *arXiv preprint arXiv:1807.05118*, 2018.

[31] "Python programming language, version 3.10," https://www.python.org/downloads/release/python-3100/, 2021, accessed: [Insert date here].

[32] S. Van Der Walt, S. C. Colbert, and G. Varoquaux, "The numpy array: A structure for efficient numerical computation," *Computing in Science & Engineering*, vol. 13, no. 2, pp. 22–30, 2011.

[33] W. McKinney, "Data structures for statistical computing in python," in *Proceedings of the 9th Python in Science Conference*, S. van der Walt and J. Millman, Eds., 2010, pp. 51–56.

[34] "Geopandas: Python tools for geographic data," https://geopandas.org, 2021, accessed: [Insert date here].

[35] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," in *Journal of Machine Learning Research*, vol. 12, 2011, pp. 2825–2830.

[36] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*, vol. 32. Curran Associates, Inc., 2019, pp. 8024–8035.

[37] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[38] University of Oslo, "Uio ai hub node project: It resources and ml nodes," https://www.uio.no/tjenester/it/forskning/kompetansehuber/uio-ai-hub-node-project/it-resources/ml-nodes/, 2024, accessed: 2024-06-10.

APPENDIX

The Appendix showcases visualizations of the correlation between LSTM cell states and meteorological features—humidity, minimum temperature, and maximum temperature—analyzed across various datasets. These correlations, presented for models trained with and without Snow Water Equivalent (SWE), illustrate the LSTM's capability to synchronize with temporal patterns of these critical variables throughout the year, highlighting its

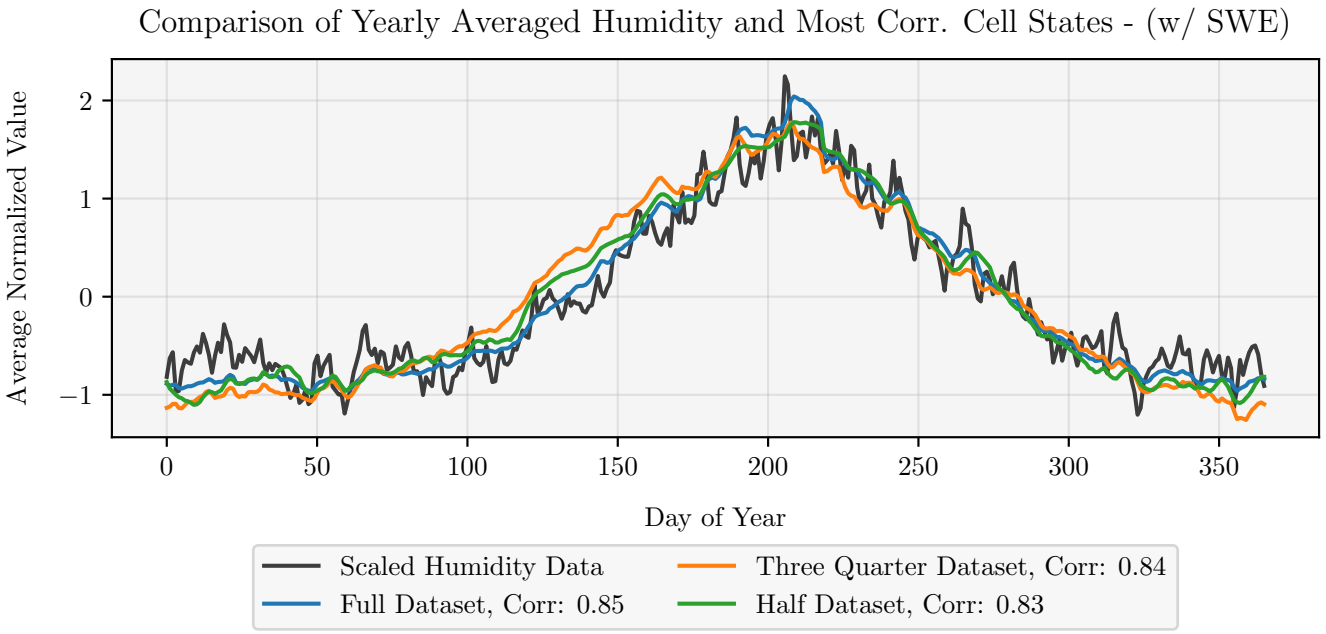

Figure 9: **Comparison of yearly averaged humidity and most correlated cell states for models trained with SWE** The black line represents the normalized feature series, while the most correlated cell states for full (NSE = 0.85), three-quarter (NSE = 0.84), and half (NSE = 0.83) dataset models are shown in blue, orange, and green, respectively.

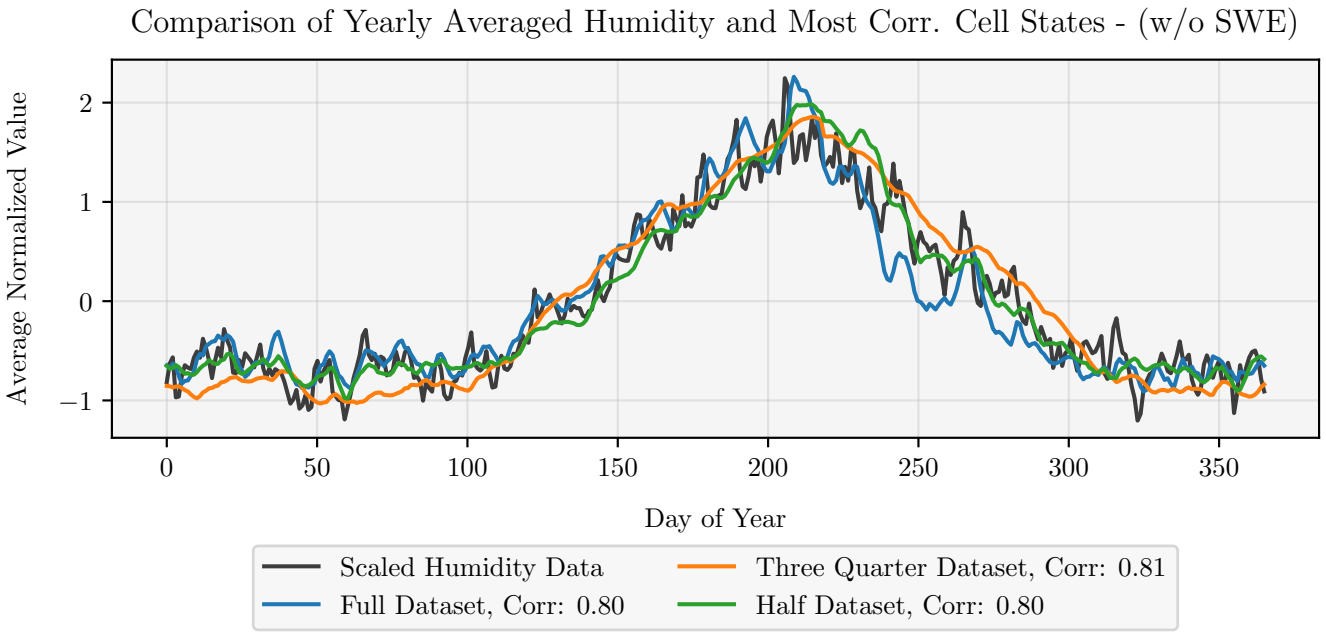

Figure 10: **Comparison of yearly average humidity and most correlated cell states for models trained without SWE.**The black line represents the normalized feature series, while the most correlated cell states for full (NSE = 0.80), three-quarter (NSE = 0.81), and half (NSE = 0.80) dataset models are shown in blue, orange, and green, respectively.

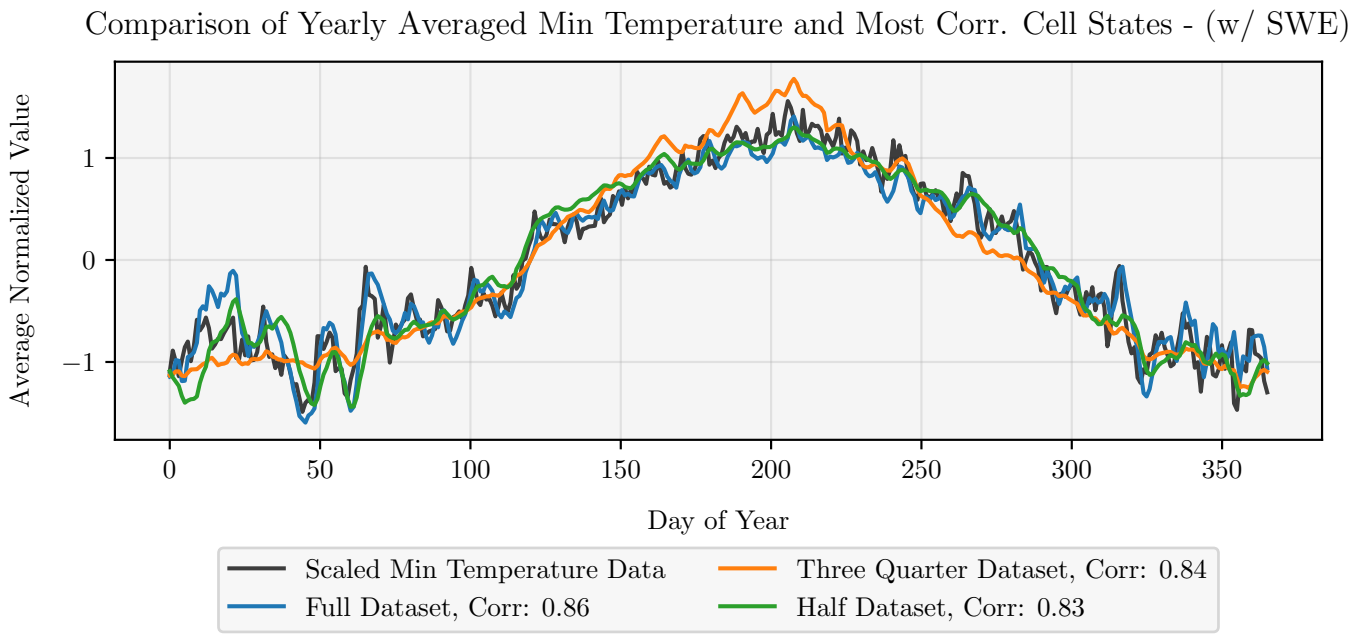

Figure 11: **Comparison of yearly minimum temperature and most correlated cell states for models trained with SWE.** The black line represents the normalized feature series, while the most correlated cell states for full (NSE = 0.86), three-quarter (NSE = 0.84), and half (NSE = 0.83) dataset models are shown in blue, orange, and green, respectively.

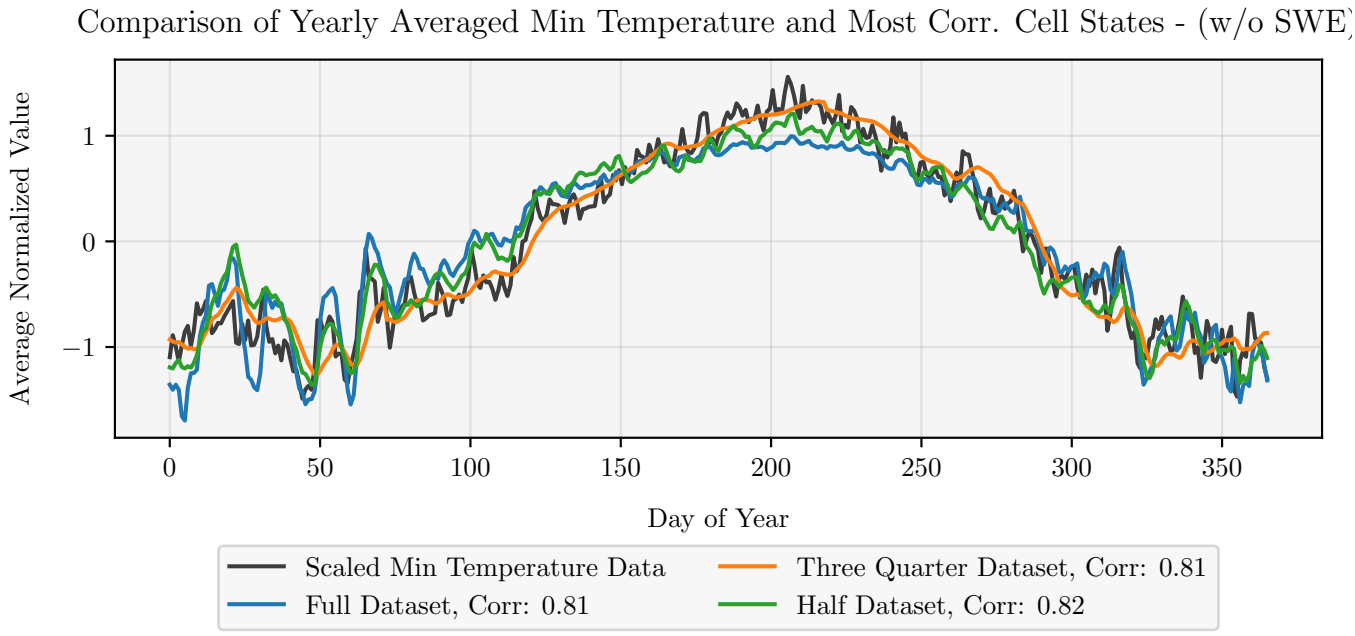

Figure 12: **Comparison of yearly average minimum temperature and most correlated cell states for models trained without SWE.**The black line represents the normalized feature series, while the most correlated cell states for full (NSE = 0.81), three-quarter (NSE = 0.81), and half (NSE = 0.82) dataset models are shown in blue, orange, and green, respectively.

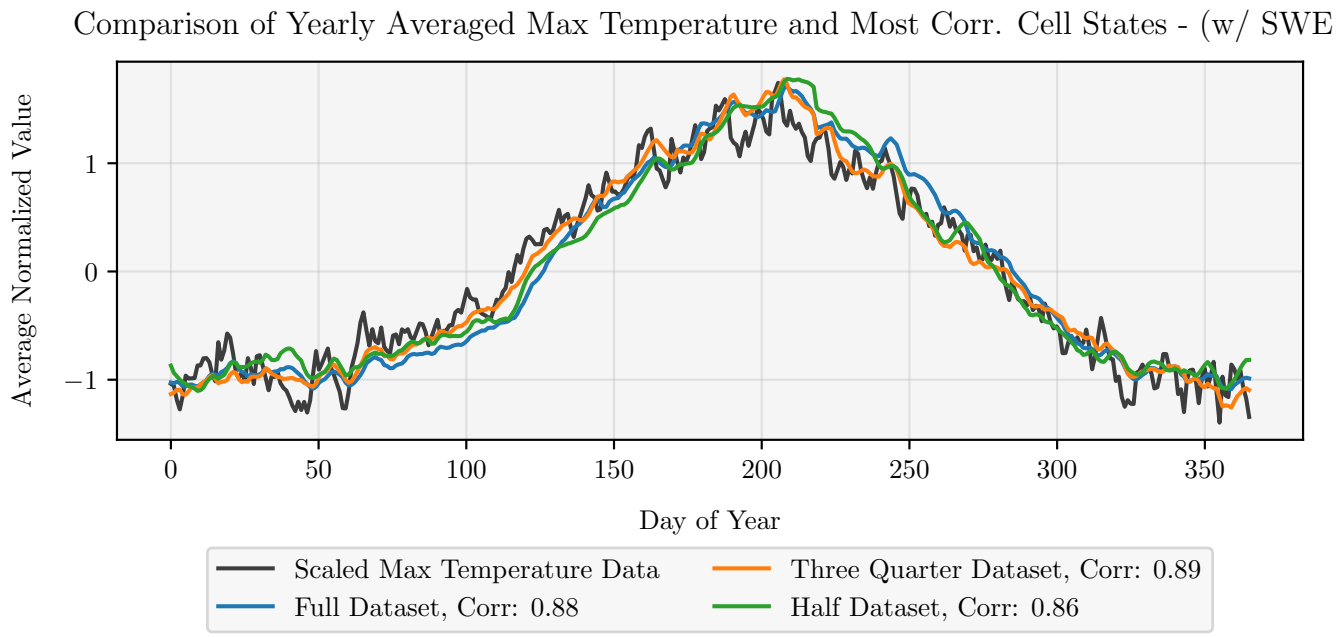

Figure 13: **Comparison of yearly average maximum temperature and most correlated cell states for model trained with SWE.**The black line represents the normalized feature series, while the most correlated cell states for full (NSE = 0.88), three-quarter (NSE = 0.89), and half (NSE = 0.86) dataset models are shown in blue, orange, and green, respectively.

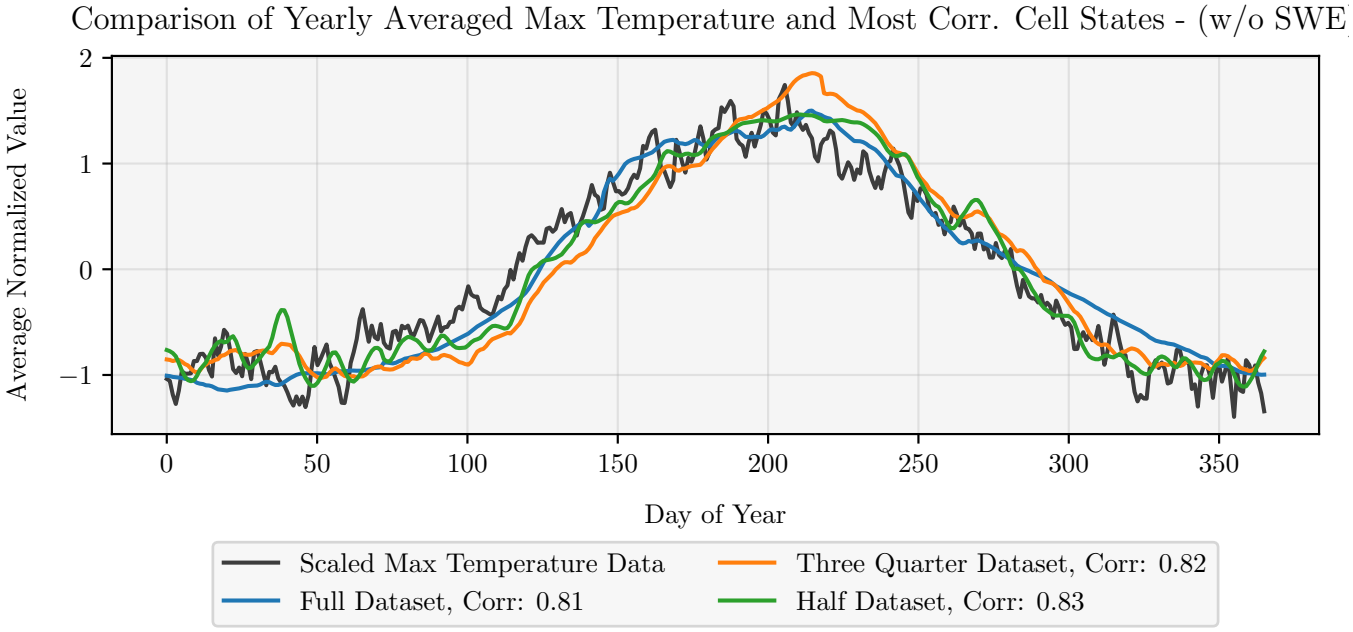

Figure 14: **Comparison of yearly average maximum temperature and most correlated cell states for models trained without SWE.**The black line represents the normalized feature series, while the most correlated cell states for full (NSE = 0.81), three-quarter (NSE = 0.82), and half (NSE = 0.83) dataset models are shown in blue, orange, and green, respectively.