Transformer neural networks for interpretable flood forecasting

Transformer neural networks for interpretable flood forecasting

Marco Castangia[a,*], Lina Maria Medina Grajales[a], Alessandro Aliberti[a], Claudio Rossi[c], Alberto Macii[a], Enrico Macii[b], Edoardo Patti[a]

[a] *Dept. of Control and Computer Engineering, Politecnico di Torino, 10129 Torino, Italy, email: {name.surname}@polito.it*
[b] *Interuniversity Dept. of Regional and Urban Studies and Planning, Politecnico di Torino, 10129 Torino, Italy, email: {name.surname}@polito.it*
[c] *Links Foundation, 10129 Torino, Italy, email: {name.surname}@linksfoundation.com*

Abstract

Floods are one of the most devastating natural hazards, causing several deaths and conspicuous damages all over the world. In this work, we explore the applicability of the Transformer neural network to the task of flood forecasting. Our goal consists in predicting the water level of a river one day ahead, by using the past water levels of its upstream branches as predictors. The methodology was validated on the severe flood that affected Southeast Europe in May 2014. The results show that the Transformer outperforms recurrent neural networks by more than 4% in terms of the Root Mean Squared Error (RMSE) and 7% in terms of the Mean Absolute Error (MAE). Furthermore, the Transformer requires lower computational costs with respect to recurrent networks. The forecasting errors obtained are considered acceptable according to the domain standards, demonstrating the applicability of the Transformer to the task of flood forecasting.

Keywords: flood forecasting, water level, deep learning, neural network, transformer, LSTM
2010 MSC: 00-01, 99-00

1. Introduction

According to the World Meteorological Organization floods are one of the most frequent and destructive natural hazards, accounting for 44% of the total number of worldwide weather-related disasters reported in the period from 1970 to 2019 [1]. Due to their increasing frequency and sever-

*marco.castangia@polito.it

ity, floods are raising great concerns for the worldwide governments. Although the occurrence of these events cannot be avoided, different measures can be taken to reduce their effects, minimize the susceptibility to damage and mitigate their impact. The problem should be approached from different sides, including infrastructure improvements, flood plan regulations, preparedness and education. These actions are enclosed in the so-called *flood management* [2]. One of the most important strategies adopted to reduce the susceptibility to damage is flood awareness and prediction. Indeed, knowing in advance the possible occurrence of floods could help people to better manage these events and rapidly respond to them. Flood prediction represents the most effective countermeasure to mitigate the damages of sudden flooding events. Indeed, the production of accurate flood forecasts allows to manage the occurrence of extreme water events beforehand their arrival. They can help the population to deal with the risks of floods in their region, thus avoiding major losses of human lives due to unpreparedness. Most importantly, local authorities can rapidly setup protection measures to rescue the population of the threatened sites. Moreover, flood forecasting is also able to quantify the magnitude of floods, thus allowing to prepare commensurate resources to face the adversities.

The problem of flood prediction has traditionally been a topic of great interest in the hydrology field. Historically, physical and numerical models represented the most widely used tools for flood forecasting [3] [4]. However, they usually require large volumes of both hydrological and geomorphological data, which are quite costly to be obtained. Moreover, the large number of parameters involved in the numerical models often result in prohibitive computational costs. In addition, large teams of experts are usually needed for the development of physical models, which further increase the development costs of such systems [5]. The advancements in computing technologies and the increasing availability of data sets from monitoring sensors determined the growing adoption of statistical models for the prediction of flooding events. As a matter of fact, statistical models require fewer input parameters with respect to numerical models and demonstrated a greater forecasting accuracy in several use cases [6] [7]. Statistical models usually approach the problem of flood prediction by forecasting a target variable, which can be either the water level of a river or the amount of rainfall measured over a certain area. In this work, we followed the former approach, using the water level of the river as an indicator of the flooding risk.

In the last decades, computing power grew significantly, allowing the generation of more advanced statistical models. Among them, the Artificial Neural Network (ANN) is one of the most

commonly used. Multiple studies have shown the superiority of these models over traditional statistical methods. In [8], a feed-forward neural network has been compared against a multiple linear regression model with the aim of predicting the daily runoff from the measurements of the previous seven days. The neural network showed superior performance with respect to its linear counterpart, motivating the use of neural networks for streamflow forecasting. Zounemat et al. [9] conducted a similar study, comparing the predictive performance of an artificial neural network with those of a linear regression model. In this case, the authors used the past runoff values of upstream stations as input. Once again, the neural network demonstrated superior performance with respect to the linear model. The effectiveness of neural networks lies in the use of non-linear activation functions, which allow to capture more complex relationships with respect to linear models.

Recently, researchers started to adopt deeper models to further enhance the performance of neural networks and model even more complex relationships [10]. Shu et al. [11] explored the applicability of convolutional neural networks (CNN) for forecasting the monthly streamflow of a river, comparing its performance with an artificial neural network and an extreme learning machine. The results showed that the CNN has superior performance with respect to the shallow neural architectures. Huang et al. [12] designed a convolutional neural network to forecast the daily streamflow of several sites based on the past water levels of their neighbors. The same CNN has been trained multiple times for each site by using a transfer learning technique. The convolutional neural network outperformed simpler approaches, including the autoregressive model, multilayer perception network and kernel ridge regression. Adikari et al. in [13] proposed a Convolutional Neural Network (CNN) for performing flood and drought forecasting in both arid and tropical regions. The results show that the CNN outperforms other models in flood forecasting and can better handle multiple features in input. Hosseiny in [14] used a more complex CNN called U-NET to predict the water depth over multiple locations surrounding the river's basin, given in input the ground elevation of the predicted points and the flooding discharge of the river.

Convolutional neural networks can also be adapted to process graph data, such as those represented by a network of gauges distributed along a river basin. Sit et al. in [15] investigated the effectiveness of a graph convolutional neural network for predicting the hourly river flow from the past water levels of upstream gauges. The results showed that graph convolutions improved upon standard convolutions, motivating the use of this new architecture. A similar approach has been adopted in other studies, demonstrating that graph convolutional neural networks are perfectly

suitable for this task thanks to their capability of handling spatio-temporal features [16, 17, 18].

Among all the different neural architectures applied to the task of flood forecasting, the Long Short-Term Memory (LSTM) neural network is certainly the most widely investigated in this field. In fact, the LSTM is specifically designed to process sequential inputs such as those described by rivers' water levels and historical rainfall's measurements. Campos et al. in [19] showed that the LSTM neural network outperforms the ARIMA model in forecasting the streamflow of some Brazilian rivers with an autoregressive approach. Differently from ARIMA, which is a linear model, the LSTM can capture more complex non linear relationships between consecutive water levels. The same conclusions can be drawn from the work of Xiang et al. [20], which shows that the LSTM-seq2seq model overcomes the performance of other linear models, including linear, lasso and ridge regression methods. When compared with other classical machine learning algorithms, the LSTM remains the preferred approach for flood forecasting. Dazzi et al. in [21] compared a LSTM neural network with a Support Vector Machine (SVM) and a Multilayer Perceptron (MLP) in predicting the hourly water levels of an Italian river from the measurements of two upstream stations. The results demonstrate that the LSTM outperforms both the SVM and the MLP by a good amount in terms of different forecasting metrics. Other related works assessed the superiority of the LSTM neural network with respect to traditional neural architectures, which definitely places the LSTM among the state-of-the-art models for flood forecasting [22] [23]

Very recently, researchers found that sequences can be better analyzed by using the so-called attention mechanism, which allows analyzing longer sequences with respect to plain recurrent networks. Ding et al. in [24] implemented a LSTM model enhanced with a spatio-temporal attention mechanism, which allows to relate the water levels of different river's sites and at different lagged timestamps to improve both accuracy and interpretability of neural networks. The authors of [25] proposed a graph convolutional neural network (GCN) for better exploiting the water levels of different stations, employing a spatio-temporal attention mechanism to further improve the accuracy of their model. In general, all the works that used a form of attention mechanism (spatial or temporal) combined with recurrent layers demonstrated that this approach outperforms conventional deep learning methods in the task of flood forecasting [26], [27], [28].

In practice, the latest advancements in language translation demonstrated that we can completely substitute recurrent layers with the self-attention mechanism, which allows to model even longer input sequences [29]. Shortly after, the research community started to apply the Trans-

former to the task of time series forecasting, overcoming the accuracy of recurrent neural networks [30, 31, 32]. In addition, the attention-mechanism allows to tackle very different tasks involving both structured and unstructured data, motivating its application in diverse predictive domains [33]. Liu et al. in [34] proposed a Transformer neural network for predicting the monthly streamflow of the Yangtze River using both past water levels and the El Niño–Southern Oscillation (ENSO) as input. The Transformer architecture demonstrated superior performance with respect to several machine learning models, including convolutional and recurrent neural networks.

In the scope of flood forecasting, the ability of producing timely predictions is crucial to implement effective countermeasures for mitigating the impact of flooding events. Indeed, a good flood forecasting system should provide the highest accuracy with the lowest computational costs. In the task of language translation, the Transformer demonstrated superior performance with respect to recurrent neural networks both in terms of accuracy and execution times. In the same way, we believe that the Transformer could also improve upon recurrent neural networks in the task of flood forecasting by increasing the forecasting accuracy and lowering the computational costs. In this paper, we propose a Transformer neural network for tackling the problem of flood forecasting. In detail, we aim at predicting the water level of a target river by using the past water levels of its upstream branches as input. Differently from [34], we applied the Transformer directly to the raw streamflow data without applying any transformation to the input (e.g. variational mode decomposition). In addition, the novelty of our approach consists in using more input variables from neighboring stations, thus truly exploiting the potentiality of the self-attention module. To demonstrate the effectiveness of our model, we compared the prediction performance of the Transformer with both GRU and LSTM neural networks. The results show that the Transformer presents superior forecasting capabilities with minor execution times, which definitely motivates its use for flood forecasting. The effectiveness of our method lies in the multi-head attention mechanism implemented in the Transformer's encoder layers. In particular, we found that the attention mechanism assigns different weights to the various inputs based on their relevance for producing accurate predictions. In this way, the Transformer attends to specific parts of the input, which allows using an even larger set of features as input. The obtained results are considered "good" according to the domain standards [35], which demonstrates the applicability of the proposed method in the real world.

The rest of this work is organized as follows. Section 2 provides a description of the data set.

Section 3 introduces the Transformer neural network and explains the architectural choices of our specific implementation. In Section 4 the Transformer is compared with recurrent neural networks and other statistical models used as benchmarks, discussing the performance in terms of accuracy and computational costs. Finally, Section 5 presents the main findings of our work and provides potential future research directions.

2. Data set

The data set used to conduct our experiments was obtained from the Hydrological Information System of the Sava River Basin (SavaHIS) [36]. The SavaHIS project handles a large fleet of hydrological and meteorological stations deployed across the Sava river and its tributaries. The Sava river passes through several countries in Southeast Europe, including Slovenia, Croatia, Bosnia & Herzegovina, Serbia and Montenegro. Here, we focused on the country of Bosnia and Herzegovina, which was severely affected by an extreme flooding event in May 2014. In particular, Doboj was one of the most severely affected cities, counting several victims due to the inundation from nearby rivers [37]. For this reason, we decided to use the water level of Doboj as our target variable to evaluate the performance of our models in a real-world scenario. In fact, the accurate estimate of future water levels can help local authorities to prepare the right countermeasures to face such catastrophic events.

To achieve this goal, we selected a set of 13 hydrological stations from the upstream branch of the river with respect to the site of Doboj. Each monitoring station provides the average water level of the river with a daily frequency, expressed as the distance in cm from a reference point. The entire monitoring period of our data set goes from 1 January 2014 to 31 December 2014, covering exactly two years of measurements without missing values. Table 1 gives some statistics of the 13 monitoring stations composing our final data set, in addition to the target station of Doboj that was used as input as well. Those statistics include the minimum, maximum and mean water level for each river, together with its standard deviation. The last column reports the distance in km of each gauge from Doboj.

Station name	Water level [cm]				Distance
	Mean	Std	Min	Max	
Doboj	-73.66	83.59	-161.00	721.00	-
Ilidža	53.45	15.75	31.00	131.00	101 km
Kalošević	55.87	28.15	24.00	367.00	17 km
Karanovac	124.20	70.65	64.00	597.00	15 km
Maglaj	109.53	72.03	28.00	948.00	21 km
Merdani	64.09	20.96	37.00	181.00	68 km
Modrac	68.54	55.72	13.00	647.00	41 km
Olovo	49.85	43.76	10.00	543.00	78 km
Raspotočje	75.29	47.18	22.00	499.00	61 km
Reljevo	148.76	43.87	103.00	460.00	96 km
Čumurija	78.58	34.00	8.00	190.00	100 km
Strašanj	63.64	90.44	-78.00	724.00	55 km
Vrelo Bosne	37.31	13.92	17.00	78.00	101 km
Zavidovići	108.42	77.68	-3.00	846.00	33 km

Table 1: Data set summary.

The data set was split into a training set and test set for evaluating and comparing the performance of our models. The training period covers roughly 66% of the entire data set, from January 2013 to April 2014 (16 months). Accordingly, the test set consists of the remaining months from May 2014 to Dec 2014 (8 months). Notice that the severe flood occurring on 13 May 2014 is contained in the test set, and it will serve to truly demonstrate the applicability of our models in a real flooding scenario. The training set was further split into an actual training set and a smaller validation set, which has been used for avoiding overfitting during the neural networks' training. The validation set covers the period from February 2014 to April 2014 (3 months), while the final training set goes from January 2013 to January 2014, included. For the sake of clarity, we reported the details of data set split in Table 2. Notice also that we respected the temporal order of the events, by using only data prior to the flood for training our models without leaking data from the future.

Training	Validation	Test
2013/01/01 - 2014/01/31	2014/02/01 - 2014/04/30	2014/05/01 - 2014/12/31

Table 2: Data set split into training, validation and test.

3. Methodology

The full pipeline of our methodology for flood forecasting is depicted in Figure 1. Notice that the training and validation phases are repeated multiple times to find the best combination of hyperparameters for our model. The final test phase instead is executed only once, in order to evaluate the forecasting performance of the Transformer on unseen data. In the following, we introduce our specific implementation of the Transformer neural network, discussing all the relevant architectural choices that conducted to the final architecture.

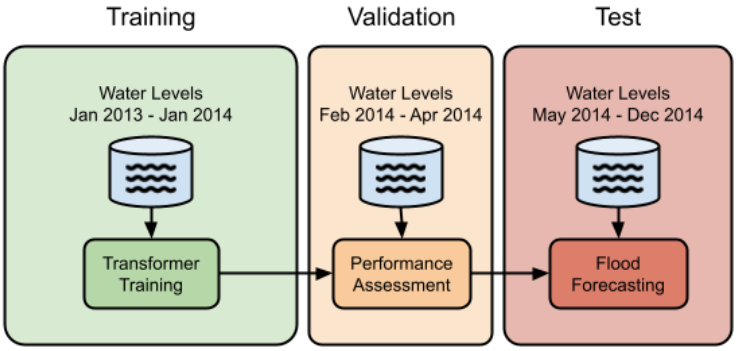

Figure 1: Pipeline of our flood forecasting approach.

3.1. Transformer

The Transformer is a neural network that was firstly introduced by Vaswasni et al. in [29] as a groundbreaking architecture for solving machine translation problems. The Transformer has been early adopted in other application domains involving the analysis of fairly long input sequences, such as time series forecasting and classification [30]. In fact, the self-attention mechanism of the Transformer completely substituted recurrent layers, allowing to analyse even longer input sequences.

The computations of the self-attention can be divided into the following steps. Firstly, each point of the input sequence is mapped to three different representations with dimension d_{model}, which are called query (Q), key (K) and value (V). Then, the Q vectors are matched against all the K vectors by means of a dot-product multiplication. The square matrix obtained in this way is scaled and passed through a softmax function to obtain the attention scores between the different points of the sequence. Finally, the attention scores are multiplied by the V vectors to generate a new representation of the input sequence. Indeed, the self-attention modifies the representation of each point by taking into account the weighted contribution of all the other points in the sequence. In this way, distant points in the sequence can attend to the value of each other, sharing important information that could be missed otherwise. The self-attention mechanism is summarized by the following formula, where d_k is the dimension of the key vectors.

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{1}$$

The multi-head attention used by the Transformer simply adds a bit of redundancy to the self-attention computation. In fact, the values of Q, K, and V are split into multiple chunks with dimension d_{model}/h, where h is the number of attention heads. Each head applies the same computations of Equation 1 to each individual chunk. After that, all the new representations are concatenated and passed through a final linear transformation. The use of the multi-head attention allows increasing the number of combinations between the different points of the input sequence, thus increasing the probability of finding relevant relationships.

Since the self-attention mechanism is completely agnostic of the order in the input sequence, we need to add some information to keep track of the position of the different points. This goal is achieved by adding a static *positional encoding* to the original input embedding of the Transformer model. The position of each point is encoded by means of the following formula, where *pos* is the position within the input sequence and i is the index of the embedding value.

$$\begin{aligned} PE_{(pos,2i)} &= sin(pos/100002i/d_{model}) \\ PE_{(pos,2i+1)} &= cos(pos/100002i/d_{model}) \end{aligned} \tag{2}$$

The Transformer used in this work is slightly different from the original implementation presented in [29]. The most important difference is the lack of decoder layers. In fact, differently from traditional machine translation problems, where sequence-to-sequence (seq2seq) models are usually

employed, in our case we are dealing with a single output value that represents the predicted water level for the next day. Therefore, we decided to employ only encoder layers and map the extracted representations directly to the output value, following the implementations of previous works from the literature [33, 38, 39]. Interestingly, we found that a single encoder layer already provides the best performance with respect to higher numbers of layers, as shown in Figure 3. Another important difference in our architecture is the lack of normalization layers. With reference to Figure 4, we found that layer normalization reduces the generalization capabilities of our model to a great extent. The last architectural choice that we made consists of the use of a final global average pooling layer before the output. As depicted in Figure 5, we found that the average pooling layer gives slightly better results than simply flattening the final representations. The full architecture of our Transformer is depicted in Figure 2. Notice that the multi-head attention module is followed by a small feed-forward neural net, which applies a non-linear transformation to the input. The d_{model} was set to 512 and the number of units in the feed-forward module is 2048, similarly to the original Transformer implementation.

The Transformer was implemented in Python by using Tensorflow and the Keras API [40]. To train the Transformer we used the adaptive moment estimation (Adam) optimization algorithm [41]. The best convergence was achieved with a very small learning rate equal to 0.00001. The loss function that was minimized during the training of the Transformer was the Mean Squared Error (MSE), defined by Equation 3. In this formula, n is the number of samples, whereas y_i and $\hat{y}_i$ are the observed and predicted values. The batch size was set to 32 samples, and the maximum number of epochs was set to 1000. To avoid overfitting, the early stopping criteria was employed, interrupting the training procedure if the model performance does not improve for more than 30 epochs. Table 3 summarizes the hyperparameters used to train the Transformer.

$$MSE = \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{n} \tag{3}$$

Hyperparameter	**Value**
Loss function	Mean squared error
Optimization algorithm	Adam
Learning rate	0.00001
Batch size	32 samples
Number of epochs	1000
Stopping criteria	Early stopping with patience of 30 epochs

Table 3: Training hyperparameters.

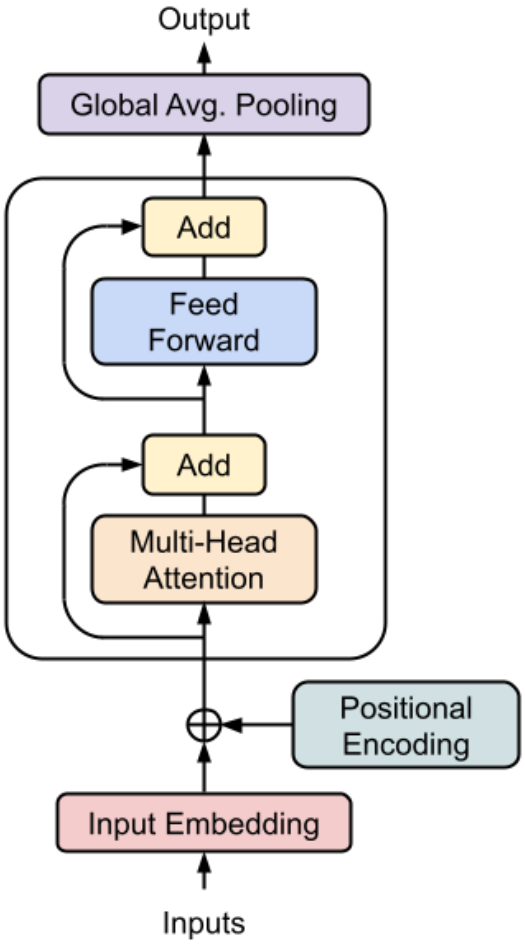

Figure 2: Transformer architecture.

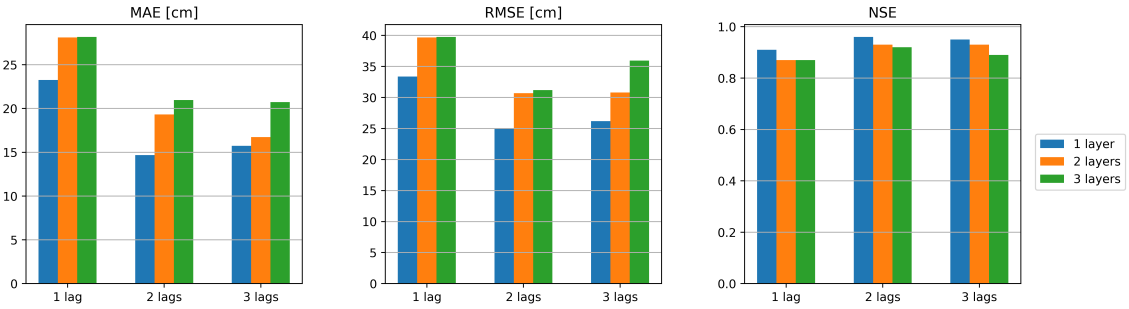

Figure 3: Performance with different number of encoder layers.

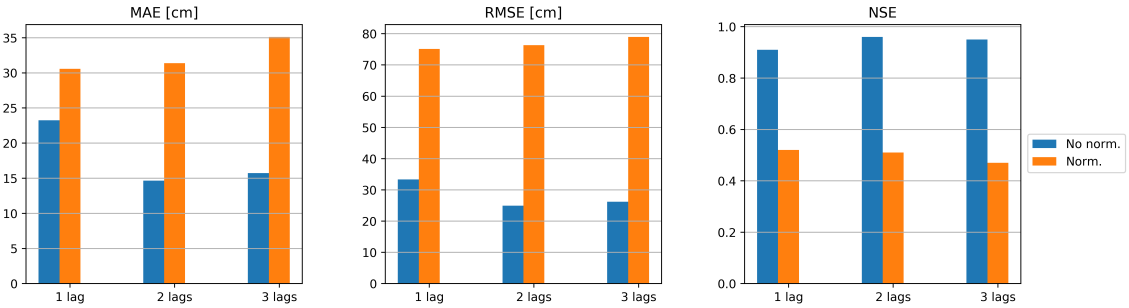

Figure 4: Performance with and without layer normalization.

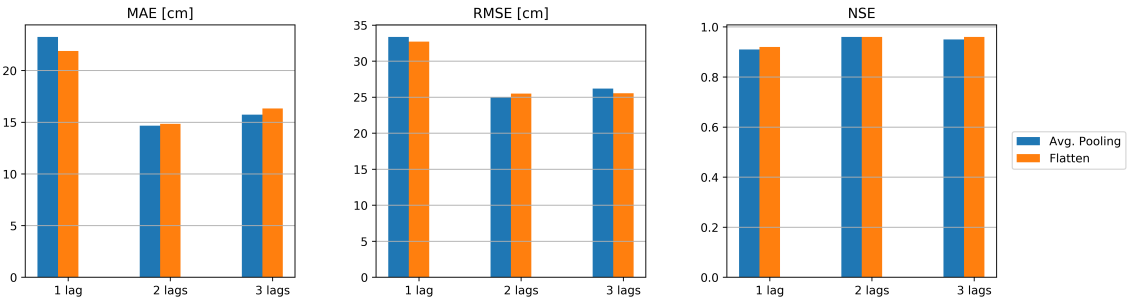

Figure 5: Performance with final global average pooling and flatten.

4. Results and discussion

This section presents the prediction results obtained with the Transformer neural network applied to the task of flood forecasting. First of all, we give a quick description of the LSTM neural

network that we used as benchmark. Then, we briefly introduce the evaluation metrics that we used to compare the performance of our models. After that, we present the forecasting performance of the two models with a varying number of lagged inputs. We also provide a general interpretation of the attention maps extracted from the Transformer neural network. To conclude, we compare the two models in terms of the required execution times for both training and testing.

4.1. Benchmark models

In Section 1, we provided some relevant works demonstrating that LSTM-based models currently represent the state of the art in flood forecasting [22]. Indeed, LSTM outperformed all traditional machine learning algorithms in this task, including shallow neural network approaches [21]. For this reason, we decided to primarily compare the forecasting performance of the Transformer with those of an LSTM-based benchmark model. In addition to the LSTM, we also decided to compare the Transformer with another recurrent neural network using the Gated Recurrent Unit (GRU) [42], which uses significantly less parameters than the LSTM. Finally, we also compared the Transformer with simpler models such as persistence and Ridge regression, to ensure that our approach at least overcomes the performance of well-known baseline models. To find the best architecture of our LSTM and GRU models, we run multiple validation tests by changing the number of recurrent layers together with their hyper-parameters. We finally opted for a stack of two recurrent layers with 128 units, using a rectified linear units (ReLU) activation function. This configuration achieved the best performance on the validation set, and was retained for the final models' comparison on the test set. For the sake of clarity, the full architecture of the recurrent neural networks is reported in Table 4. The Ridge regression model achieved the best performance with a very small regularization factor (i.e. 0.0001), which is practically equal to training a simple linear regression model without applying any regularization.

Layer	**Activation**	**Units**	**Return sequence**
LSTM/GRU	ReLU	128	True
LSTM/GRU	ReLU	128	False

Table 4: Architecture of the LSTM and GRU benchmark models.

4.2. Evaluation metrics

To evaluate the forecasting performance of the Transformer, we adopted four statistical indicators commonly used for the evaluation of hydrological models [35]. The evaluation metrics are the Mean Absolute Error (MAE), the Root Mean Square Error (RMSE), the Nash-Sutcliffe efficiency coefficient (NSE) and the Mean Bias Error (MBE). The four metrics are defined by the following equations, where y is the vector of the measured water levels, $\hat{y}$ contains the predicted water levels, $\bar{y}$ is the mean value of the measured water levels and n is the number of samples. Notice that a good set of predictions present an NSE closer to 1, while for the MAE, MBE and RMSE follow the rule of the lower the better. The MBE was only computed for top 2% peak levels of the test set in order to evaluate the effectiveness of each model in predicting exceptional water levels and flooding events. The MBE is a useful metric to understand the extent to which the model is underestimating or overestimating the actual water levels.

$$MAE = \frac{\sum_{i=1}^{n} |y_i - \hat{y}_i|}{n} \tag{4}$$

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{n}} \tag{5}$$

$$NSE = 1 - \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y}_i)^2} \tag{6}$$

$$MBE = \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)}{n} \tag{7}$$

4.3. Model performance

Table 5 reports the forecasting performance of the Transformer compared to the benchmark models in terms of MAE, RMSE and NSE. In addition, we also reported the MBE computed on the top 2% peak levels of the test set. For each statistical model we provided the final results with three different lagged inputs, because we noticed that different models may perform better depending on the number of previous days used as input. For the persistence, which uses only the previous day value, we just reported the results with a lag of one day. For completeness, we also reported the

results obtained on the training and validation sets, in addition to those of the final test set. Table 5 shows that the Transformer obtained the best performance in terms of all the reported metrics. The highlighted values indicate the model with the lowest error on the test set. Notice that the Transformer achieved the lowest errors with a lagged input of two days. In comparison with the best performance of recurrent neural networks, the Transformer improved the RMSE by 4% and reduced the MAE by more than 7%. Interestingly, Ridge regression achieved a lower MAE than recurrent neural networks, which motivates its use as a valid alternative to more complex neural models. The persistence was largely outperformed by all models using more than one lagged input, showing that previous values from more than one day in the past are important for the forecasting results. According to the hydrology forecasting standards reported in [35], the Transformer presents "acceptable" performance, with an RMSE lower than 0.83 standard deviations and an NSE greater than 0.65. In practice, the performance can be definitely considered as "good", since our results are closer to the threshold of 0.45 standard deviations and the NSE is greater than 0.8. In fact, we can notice an increase in the RMSE between the validation set and the test set, from 25.84 cm to 38.28 cm for a lagged input of two days. The cause of this discrepancy lies in the flooding event of May 2014, that showed water levels never seen before by the model. However, the Transformer successfully predicted also the water levels during the flooding event, as depicted in Figure 6. This fact is also supported by the lowest MBE (in absolute value) obtained by the Transformer with respect to all the other models, showing that the Transformer is a valid model for predicting future unexpected water levels in this region."

Lagged Inputs	**Models**	**Training**			**Validation**			**Test**			
		MAE [cm]	**RMSE** [cm]	**NSE** [–]	**MAE** [cm]	**RMSE** [cm]	**NSE** [–]	**MAE** [cm]	**RMSE** [cm]	**NSE** [–]	**MBE** [cm]
1 day	Persistence	13.08	30.16	0.83	17.44	34.03	0.89	26.84	63.80	0.72	-127.80
	Ridge	8.59	14.73	0.96	14.72	25.14	0.94	27.33	53.66	0.80	-206.04
	GRU	11.54	15.36	0.96	13.93	26.67	0.93	24.23	49.60	0.83	-220.76
	LSTM	17.44	23.98	0.87	16.66	30.93	0.90	24.21	50.88	0.82	-228.89
	Transformer	10.79	16.85	0.95	13.80	26.28	0.93	27.06	47.57	0.84	-186.12
2 days	Ridge	7.63	12.95	0.97	13.54	24.48	0.94	21.20	44.27	0.86	-131.62
	GRU	7.71	11.78	0.97	14.49	29.87	0.91	19.64	42.02	0.88	-127.14
	LSTM	11.08	17.09	0.94	16.68	30.07	0.91	20.43	40.14	0.89	-148.75
	Transformer	8.75	12.92	0.97	14.11	25.84	0.93	**18.16**	**38.28**	**0.90**	**-117.66**
3 days	Ridge	7.49	12.31	0.97	13.43	24.18	0.94	19.06	43.36	0.87	-143.66
	GRU	9.47	13.99	0.96	15.73	28.64	0.92	21.63	43.99	0.87	-118.22
	LSTM	12.03	18.24	0.94	14.86	29.54	0.91	19.66	42.21	0.88	-188.56
	Transformer	8.36	12.33	0.97	14.04	25.57	0.93	19.32	39.22	0.89	-165.55

Table 5: Comparison of models' performance.

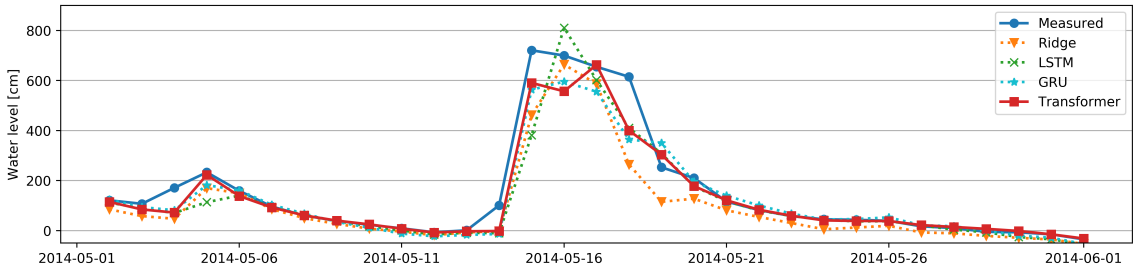

Figure 6: Water levels predicted by the Transformer on May 2014 with a prediction horizon of one day, compared with the predicted water levels of the LSTM benchmark model.

4.4. Interpretation of the attention maps

The attention scores computed within the multi-head attention layers usually reveal some hidden correlations captured by the Transformer at inference time. To demonstrate the presence of these hidden patterns, in Figure 7 we reported the attention maps of the 8 heads of our Transformer in the period from 11 May 2014 to 20 May 2014. First of all, it is interesting to notice how all the attention maps suddenly change at the start of the flooding event (14 May 2014), and then rapidly restore their previous patterns at the end (18 May 2014). Figure 8 provides a more detailed view of the 8 attention maps captured by the Transformer on 14 May 2014, where each column in the attention maps correspond to a specific upstream station, except for the first column which is the target site (Doboj). The attention maps clearly show that the Transformer applies a sort of feature selection on the input stations, giving more importance to the most relevant variables at the occurrence. This phenomenon can be evinced from the vertical patterns observable in the attention maps of Figure 7, which tend to give more weight to certain input features with respect to others. For example, Figure 8 shows that in the day preceding the flood (14 May 2014) the Transformer changed its attention maps to give more importance to the stations of Reljevo (see heads 4 and 6) and Olovo (see heads 1, 2, 3, 5, 7 and 8). With reference to Figure 9 instead, we can see that those stations anticipate to a certain extent the flooding event that affected Doboj on the following day (15 May 2014). Therefore, we can say that the Transformer correctly attends with more attention those upstream stations that exhibit a good predictive potential. However, we can also notice that other seemingly relevant upstream stations were completely ignored by the attention maps. To explain this behaviour, we can suppose that the Transformer tries to suppress redundant information, such as those existing between highly correlated upstream stations.

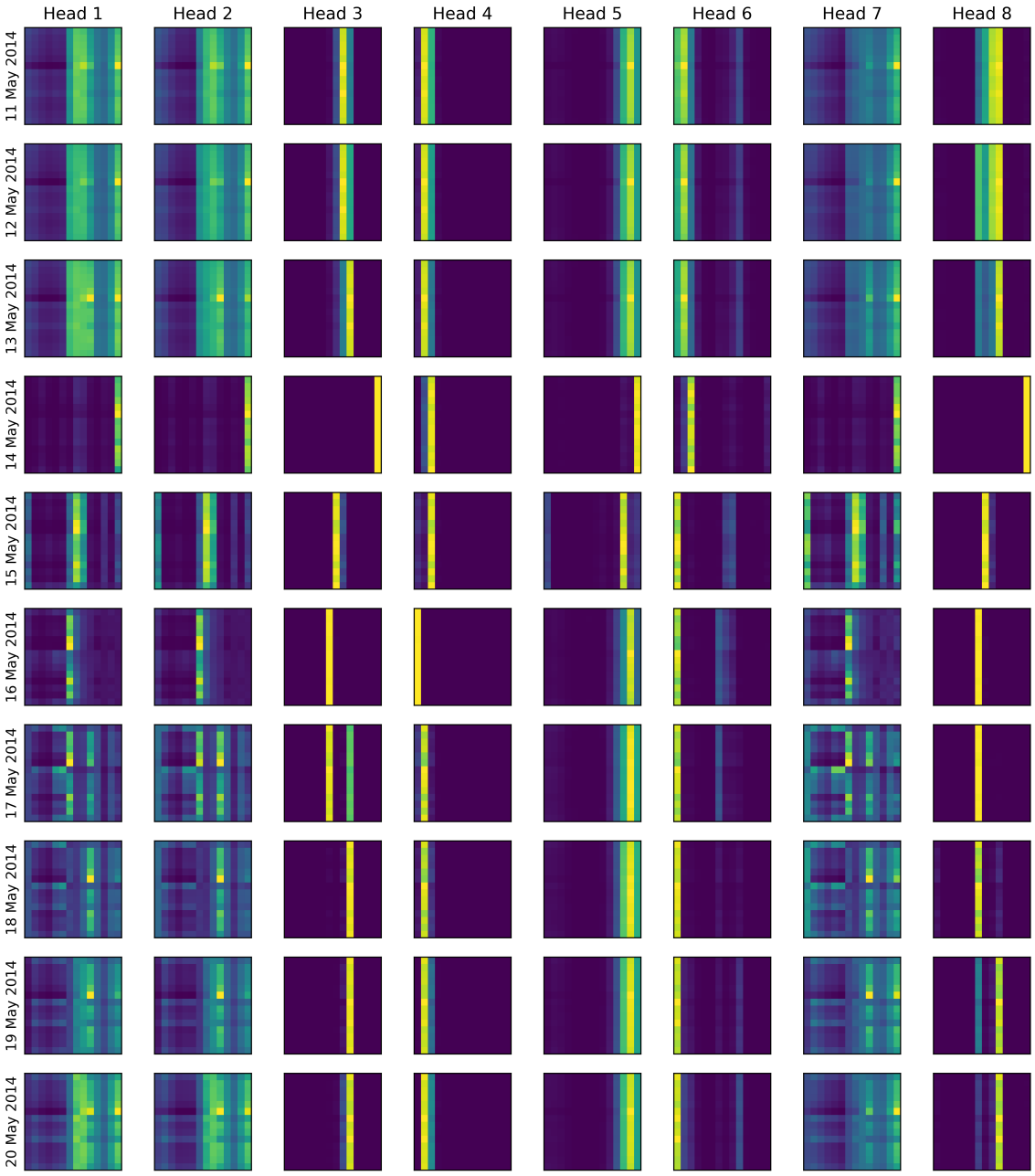

Figure 7: Attention maps from 11 May 2014 to 20 May 2014.

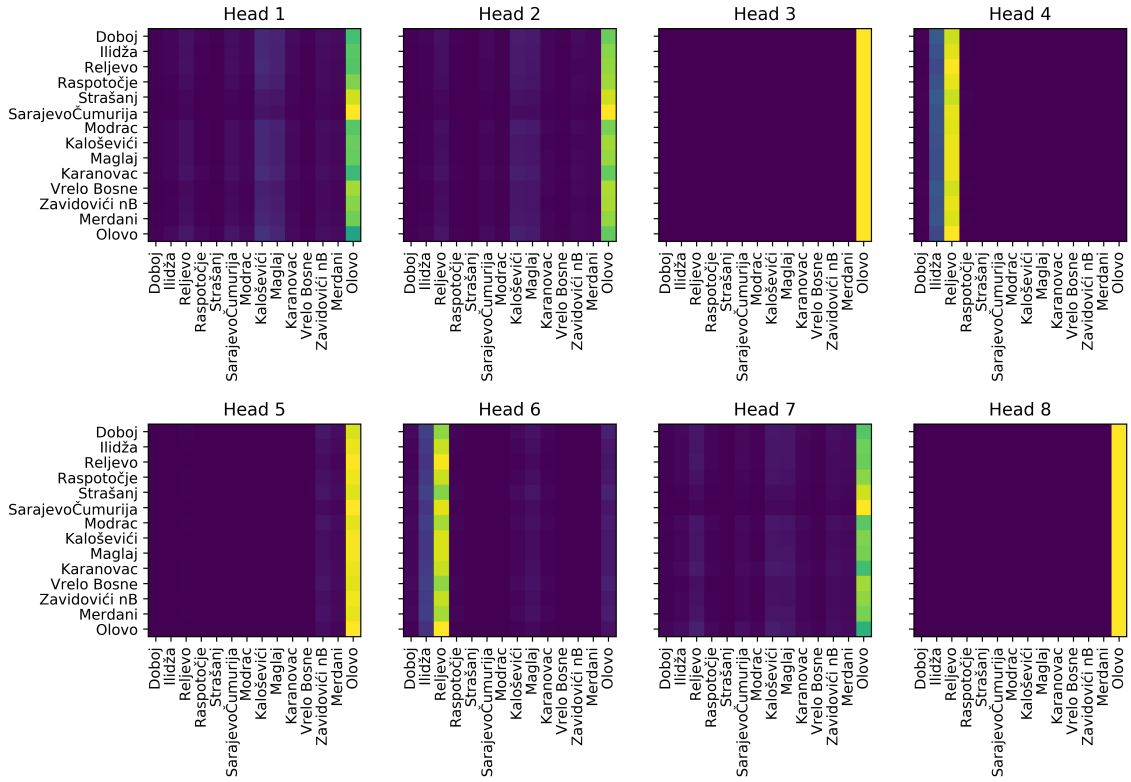

Figure 8: Attention maps on 14 May 2014.

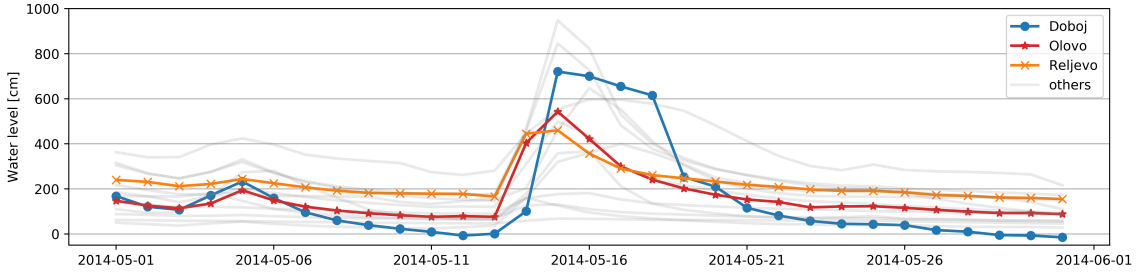

Figure 9: Water levels of the most relevant upstream stations according to the attention maps extracted on 14 May 2014.

4.5. Computational costs

Table 6 reports a comparison between the execution times and the number of parameters of the different neural networks for both training and test. All our tests were run on a workstation

equipped with a single NVIDIA GeForce RTX™ 3080 graphical processing unit and an AMD Ryzen® 9 3900 processor. The execution times are very low in both cases because of the relatively small size of our data set, which counts only 396 training samples and 244 test samples. However, we must consider that these models can be employed to monitor several hydrological stations. Moreover, they can be potentially used to forecast water levels at shorter lead times such as minutes and hours. Given these considerations, even a small percentage reduction in the execution times can result in a great improvement in the scalability of a real-world flood management system. Table 6 shows that the Transformer required on average 24 seconds for training and 0.17 seconds for testing, while the GRU (which is faster than LSTM) required 36 seconds and 0.22 seconds for the same tasks, which is roughly 50% more time for training and 29% more time for inference, respectively. In fact, recurrent networks showed longer execution times despite using significantly less parameters than the Transformer. The performance advantage of the Transformer lies in the parallel computation of the input sequences, which is not possible in the case of recurrent neural networks. In conclusion, the comparison demonstrates that the Transformer can achieve better forecasting performance with shorter execution times with respect to recurrent networks.

	Training	**Test**	**Parameters**
GRU	36 s	0.22 s	**38k**
LSTM	47 s	0.27 s	198k
Transformer	**24 s**	**0.17 s**	3M

Table 6: Execution times and number of parameters of neural networks for training and test.

5. Conclusion

In this paper, we applied the Transformer neural network for predicting potential floods with a lead time of one day ahead. To achieve this goal, we trained our model to predict the water level of a river by using past observations from its upstream stations. According to the results, the Transformer presents superior forecasting performance with respect to both LSTM and GRU recurrent neural networks, which were used as benchmark in our experiments. The obtained indicators are considered good for the standards of flood forecasting, with a Mean Absolute Error of 18 cm and a Root Mean Square Error of 38 cm for one day ahead predictions. The Nash-Sutcliffe

Efficiency coefficient is good as well, with a score of 0.90. The Transformer improved the forecasting performance by more than 4% with respect to the recurrent benchmark models. In addition, the Transformer shows shorter execution times for both training and test, because it avoids the use of recurrent layers in its architecture. The attention maps reveal that the Transformer selects the most relevant input for the target at hand, changing the attention weights based on the specific conditions on the upstream stations. As future work, we want to train the Transformer with a larger set of input features, including rainfall observations from nearby meteorological stations and other physical properties of the ground. In addition, we will combine the Transformer with Graph Neural Networks to better model both spatial and temporal information of the data set.

Acknowledgement

This work was partially supported by Shelter, which is an Horizon 2020 project funded by the European Union (grant agreement No. 821282).

References

[1] World Meteorological Organization, WMO atlas of mortality and economic losses from weather, climate and water extremes (1970–2019), `https://library.wmo.int/index.php?lvl=notice_display&id=21930#.YZX3-tCZNPY` (2015).

[2] World Meteorological Organization, Integrated flood management - concept paper, `https://www.floodmanagement.info/publications/concept_paper_e.pdf` (2009).

[3] G. K. Devia, B. P. Ganasri, G. S. Dwarakish, A review on hydrological models, Aquatic procedia 4 (2015) 1001–1007.

[4] T. H. Lee, K. P. Georgakakos, Operational rainfall prediction on meso-γ scales for hydrologic applications, Water Resources Research 32 (4) (1996) 987–1003.

[5] A. Mosavi, P. Ozturk, K.-w. Chau, Flood prediction using machine learning models: Literature review, Water 10 (11) (2018) 1536.

[6] J. Abbot, J. Marohasy, Input selection and optimisation for monthly rainfall forecasting in queensland, australia, using artificial neural networks, Atmospheric Research 138 (2014) 166–178.

[7] K. Aziz, A. Rahman, G. Fang, S. Shrestha, Application of artificial neural networks in regional flood frequency analysis: a case study for australia, Stochastic environmental research and risk assessment 28 (3) (2014) 541–554.

[8] I. Aichouri, A. Hani, N. Bougherira, L. Djabri, H. Chaffai, S. Lallahem, River flow model using artificial neural networks, Energy Procedia 74 (2015) 1007–1014.

[9] M. Zounemat-Kermani, O. Kisi, T. Rajaee, Performance of radial basis and lm-feed forward artificial neural networks for predicting daily watershed runoff, Applied Soft Computing 13 (12) (2013) 4633–4644.

[10] M. Sit, B. Z. Demiray, Z. Xiang, G. J. Ewing, Y. Sermet, I. Demir, A comprehensive review of deep learning applications in hydrology and water resources, Water Science and Technology 82 (12) (2020) 2635–2670.

[11] X. Shu, W. Ding, Y. Peng, Z. Wang, J. Wu, M. Li, Monthly streamflow forecasting using convolutional neural network, Water Resources Management 35 (15) (2021) 5089–5104.

[12] C. Huang, J. Zhang, L. Cao, L. Wang, X. Luo, J.-H. Wang, A. Bensoussan, Robust forecasting of river-flow based on convolutional neural network, IEEE Transactions on Sustainable Computing 5 (4) (2020) 594–600.

[13] K. E. Adikari, S. Shrestha, D. T. Ratnayake, A. Budhathoki, S. Mohanasundaram, M. N. Dailey, Evaluation of artificial intelligence models for flood and drought forecasting in arid and tropical regions, Environmental Modelling & Software 144 (2021) 105136.

[14] H. Hosseiny, A deep learning model for predicting river flood depth and extent, Environmental Modelling & Software 145 (2021) 105186.

[15] M. Sit, B. Demiray, I. Demir, Short-term hourly streamflow prediction with graph convolutional gru networks, arXiv preprint arXiv:2107.07039.

[16] G. Liu, S. Ouyang, H. Qin, S. Liu, Q. Shen, Y. Qu, Z. Zheng, H. Sun, J. Zhou, Assessing spatial connectivity effects on daily streamflow forecasting using bayesian-based graph neural network, Science of The Total Environment (2022) 158968.

[17] Y. Liu, G. Hou, F. Huang, H. Qin, B. Wang, L. Yi, Directed graph deep neural network for multi-step daily streamflow forecasting, Journal of Hydrology 607 (2022) 127515.

[18] A. Y. Sun, P. Jiang, M. K. Mudunuru, X. Chen, Explore spatio-temporal learning of large sample hydrology using graph neural networks, Water Resources Research 57 (12) (2021) e2021WR030394.

[19] L. C. D. Campos, L. Goliatt da Fonseca, T. L. Fonseca, G. D. d. Abreu, L. F. Pires, Y. Gorodetskaya, Short-term streamflow forecasting for paraíba do sul river using deep learning, in: EPIA Conference on Artificial Intelligence, Springer, 2019, pp. 507–518.

[20] Z. Xiang, J. Yan, I. Demir, A rainfall-runoff model with lstm-based sequence-to-sequence learning, Water resources research 56 (1) (2020) e2019WR025326.

[21] S. Dazzi, R. Vacondio, P. Mignosa, Flood stage forecasting using machine-learning methods: a case study on the parma river (italy), Water 13 (12) (2021) 1612.

[22] H. Han, C. Choi, J. Jung, H. S. Kim, Deep learning with long short term memory based sequence-to-sequence model for rainfall-runoff simulation, Water 13 (4) (2021) 437.

[23] C. Hu, Q. Wu, H. Li, S. Jian, N. Li, Z. Lou, Deep learning with a long short-term memory networks approach for rainfall-runoff simulation, Water 10 (11) (2018) 1543.

[24] Y. Ding, Y. Zhu, J. Feng, P. Zhang, Z. Cheng, Interpretable spatio-temporal attention lstm model for flood forecasting, Neurocomputing 403 (2020) 348–359.

[25] J. Feng, Z. Wang, Y. Wu, Y. Xi, Spatial and temporal aware graph convolutional network for flood forecasting, in: 2021 International Joint Conference on Neural Networks (IJCNN), IEEE, 2021, pp. 1–8.

[26] Z. Liu, W. Xu, J. Feng, S. Palaiahnakote, T. Lu, et al., Context-aware attention lstm network for flood prediction, in: 2018 24th international conference on pattern recognition (ICPR), IEEE, 2018, pp. 1301–1306.

[27] Y. Wu, Y. Ding, Y. Zhu, J. Feng, S. Wang, Complexity to forecast flood: Problem definition and spatiotemporal attention lstm solution, Complexity 2020.

[28] L. Yan, C. Chen, T. Hang, Y. Hu, A stream prediction model based on attention-lstm, Earth Science Informatics 14 (2) (2021) 723–733.

[29] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, in: Advances in neural information processing systems, 2017, pp. 5998–6008.

[30] S. Li, X. Jin, Y. Xuan, X. Zhou, W. Chen, Y.-X. Wang, X. Yan, Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting, Advances in Neural Information Processing Systems 32 (2019) 5243–5253.

[31] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, W. Zhang, Informer: Beyond efficient transformer for long sequence time-series forecasting, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 35, 2021, pp. 11106–11115.

[32] N. Wu, B. Green, X. Ben, S. O'Banion, Deep transformer models for time series forecasting: The influenza prevalence case, arXiv preprint arXiv:2001.08317.

[33] A. Jaegle, F. Gimeno, A. Brock, O. Vinyals, A. Zisserman, J. Carreira, Perceiver: General perception with iterative attention, in: International conference on machine learning, PMLR, 2021, pp. 4651–4664.

[34] C. Liu, D. Liu, L. Mu, Improved transformer model for enhanced monthly streamflow predictions of the yangtze river, IEEE Access.

[35] A. Ritter, R. Munoz-Carpena, Performance evaluation of hydrological models: Statistical significance for reducing subjectivity in goodness-of-fit assessments, Journal of Hydrology 480 (2013) 33–45.

[36] I. S. R. B. Commission, Savahis, `https://savahis.org/his`, accessed: 2021-10-10 (2021).

[37] V. Novosti, Rs: Names of victims, reported disappearance of seven people, `https://www.novosti.rs/vesti/naslovna/republika_srpska/aktuelno.655.html:492132-Vrh-poplavnog-talasa-Save-prosao-RS-bez-novih-poplava` (2014).

[38] S. Huang, D. Wang, X. Wu, A. Tang, Dsanet: Dual self-attention network for multivariate time series forecasting, in: Proceedings of the 28th ACM international conference on information and knowledge management, 2019, pp. 2129–2132.

[39] Y. Hu, F. Xiao, Network self attention for forecasting time series, Applied Soft Computing (2022) 109092.

[40] F. Chollet, et al., Keras, (Accessed on: 2022-04-01) (2015).
URL `https://keras.io`

[41] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization (2017). `arXiv:1412.6980`.

[42] K. Cho, B. Van Merriënboer, D. Bahdanau, Y. Bengio, On the properties of neural machine translation: Encoder-decoder approaches, arXiv preprint arXiv:1409.1259.