Watercourse behavior modeling using sequence prediction networks

Federico Bertani
University of Bologna
federico.bertani@studio.unibo.it

ABSTRACT

Every year million of people are involved in floods resulting in property damage and loss of human lives. Accurate prediction is therefore essential to be able to take countermeasures. In recent years traditional analytical methods for hydrological quantities forecasting has been outclassed by machine learning techniques. This report investigates the performance of four machine learning models: Feed Forward, Long-short memory, Temporal convolutional and Attention models, on their ability to learn the relation between rainfall and subsequent river level surge.

1 Introduction

Watercourses are complex systems but the main cause for their sudden alteration is rainfall. This is the assumption on which this work is based. But still the exact relation from these two entities is hard to determine. This is dependent on multiple variables like: size of the drainage basin, where and how much it rained, soil characteristic and so on. Usually a hydrological model is built specifically for a single river and require lots of effort of parametrization. [5] What this study want to research is whether by machine learning is possible to have in short time a prediction model starting from a dataset with a few features. In this way this model can be retrained for different stream gauges by simply changing the training dataset.

The main contribution of this report are summarized as it follows:

- The description of a hydrological prediction framework independent of the chosen model. This framework will be later specialized for each machine learning model used. This framework tries to make the best use, from the physical point of view, of the available features.
- The description of the 4 models used: Feed Forward, Long-short memory, Temporal convolutional and attention based. For each model a theoretical introduction is provided. Finally, the results are exposed for a comparison between the models.

2 Previous works

The ASCE Task Commitee in 2000 did one of the first works [5] on application of Artificial Neural Network to the field of hydrology. This papers shows different previous findings. As architecture, feedforward 3 layer networks are mostly used but also some basic recurrent networks. An interesting aspect is that not always the networks is regressive directly on the hydrographic quantity but for example, as done in the work of Smith and Eli (1995), regression is done on 21 coefficients of a Fourier series that models the temporal trend of hydrograph observations. A literature review done by Mosavi, Ozturk, Chau [1] exists on the usage of machine learning methods for flood prediction but it doesn't evaluate any usage of LSTM or Transformers. Chen et al. (2020) [7] pointed out the importance of self-attention for the exploitation of information in short lag-time. Ding et al. (2020) [8] gave and initial interpretation of spatial and temporal attention weight applied on flood forecasting with an LSTM.

3 Methodology

The Emilia Romagna region and specifically ARPAE (Agenzia Regionale per la Prevenzione, Ambiente e Energia) provides a webapp for downloading geo-data. [4] River discharge and level data is available. Data is offered in half-hourly, hourly and daily intervals. Half-hour frequency has been chosen. Initially discharge was used as a feature to predict but there is lack of data for this feature since the instrument to measure it is more complex and expensive than the instrument to measure river level. So it has been preferred to switch to predict river level, where the quantity of data available both in number of stations and historical data is much larger. Another data input to the network is the amount of rain, measured in kg/m^2. Finally, the last input feature is the day of the year [0-365] when the measurement was made, so that the network can learn about seasonal variations. Two basis were chosen for this study: Enza and Reno river. The second one larger than the first. First the experiments were using the Enza dataset where the distance between the hydrograph and the pluviometer is 4km. After, to increase problem hardness the experiments were moved to Reno dataset where distance is 17km. The position of the Enza and Reno river

Fig. 1. Enza and Reno river position in North Italy

in northern Italy can be seen in the figure 3. Data has been cleaned by instrument noise with a moving average with window size of 6 (3 hours). Data scaling has been applied to a feature range of [0-1]. Mean squared error is the loss function. Additional evaluation metrics are maximum absolute error and the mean absolute error. With multiple prediction steps the metric is the value of this for the last step.
The machine learning library used is Keras. [14]
For hyperparameter optimization one *first iteration* has been done throught bayesian optimization using a library created by PhD F. Nogueira. [9]

4 Hydrological framework

All the models has been used inside a hydrological framework that helps the network to learn to use properly the feature given from the physics point of view. Initially this was not used and was left to the network how to combine feature information, but a significant improvement has been seen in the results since it has been started to use this new approach.

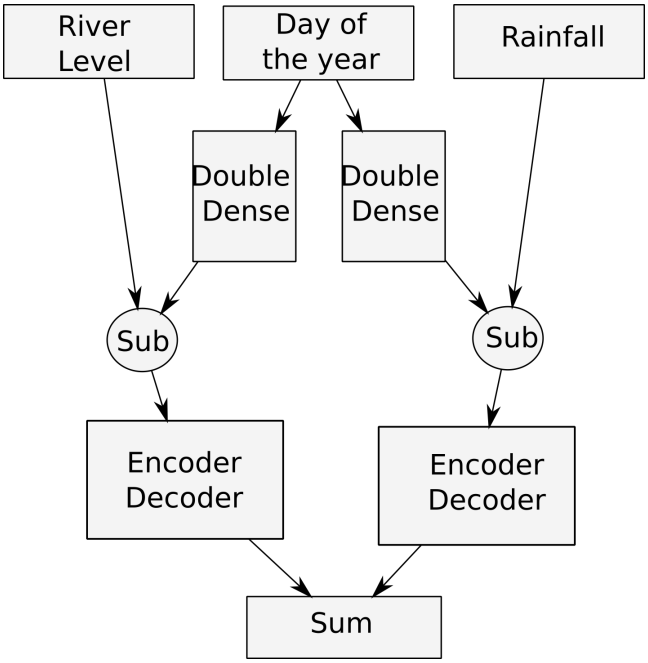

Fig. 2. Hydrological machine learning framework

A general scheme of the framework can be seen in figure 2. The day of the year is used to calculate 2 vectors: one that is subtracted to the river level to simulate the natural evaporation and drying during summer season, one that is subtracted to the rain to simulate the greater absorption of this by the ground during summer season.
Historical river level data and rainfall are first compute separately to find a different feature encoding then once the sample is encoded, is decoded to find the best forecasting of future state. This results in two vectors of the same dimension. This dimension is equal to how many half-hour is desired to forecast ahead. Lastly the river and the rain vectors are summed up. Figure 3 shown a first simple example for encoder-decoder network component with simple Dense layer as encoder. All the next ones will have the same input and output tensor shapes.

5 LSTM

Recurrent networks are used often in sequence to sequence prediction because of their ability to capture temporal relationship. In our case capturing temporal correlations is fundamental since there is always an increase of river level some hours after a rainfall. Bengio et al. (1994) [11] have shown that the traditional RNN can hardly remember sequences with a length of over 10. For daily stream flow modelling, this would imply that we could use only the last 10 steps of meteorological data as input to predict the stream flow of the next day. This period is too short for our case 1 step is equivalent to half-hour. It has been mesured in the experiment dataset that the mean time between a rainfall and a surge in river level can be up to 15 hours, so 30 steps. Long Short Term Memory (LSTM) is a modified version of recur-

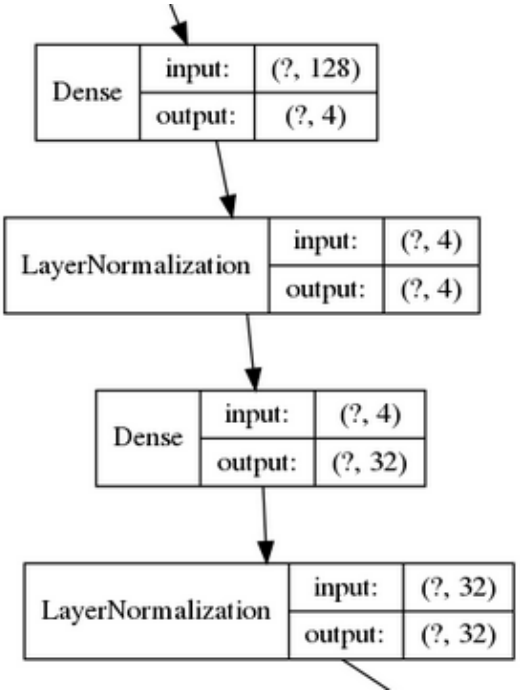

Fig. 3. Simple Encoder-Decoder with Dense layers

rent neural networks proposed by Hochreiter and Schmidhuber. [10], which is proposed to solve the problem of long-distance (time) dependence. Kratzert et.al (2018) [12] explicitly made a comparison between traditional RNN and LSTM on discharge prediction. The basin was snow influenced, so the correlation between snow melting and change in water system was very long distanced. The experiment results showed a relative error of the RNN of over 50% in melting season over LSTM.

LSTM Gates help capture long-term as well as short- term dependencies of input time series data by preventing the gradient diminishing or exploding problem. The key of LSTM to realize long-term memory lies in keeping the input information of each step in the memory unit. The hidden layer state of each output contains has access to all input information before the current moment. Since the hidden layer state is usually represented by a vector of a certain length, the network gradually compresses all the information as time goes by. Figure 4 shows the structure of an LSTM cell.

6 Temporal Convolutional Networks

Temporal Convolutional Networks were introduced in 2016 by Lea et al. [16]. TCNs are one-dimensional causal dilated Convolutional Neural Networks. When dealing with CNNs, "causal" means that each ouput y_t of the network is exclusively a function of previous input timestep $x_0,...,x_t$, while "dilated" mean that each hidden layer is given a *dilatation factor* d which controls which timestep from the previous layer the convolution is actually applied over. Given a sequence $x = x_0,..,x_n$ and a filter $f : 0,...,k-1 \rightarrow \mathbb{R}$, the *dilated causal convolution* operation $x *_d f$ on the t-th element

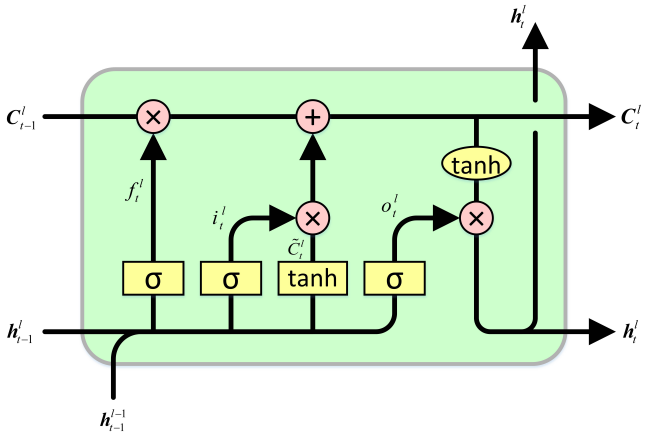

Fig. 4. LSTM cell

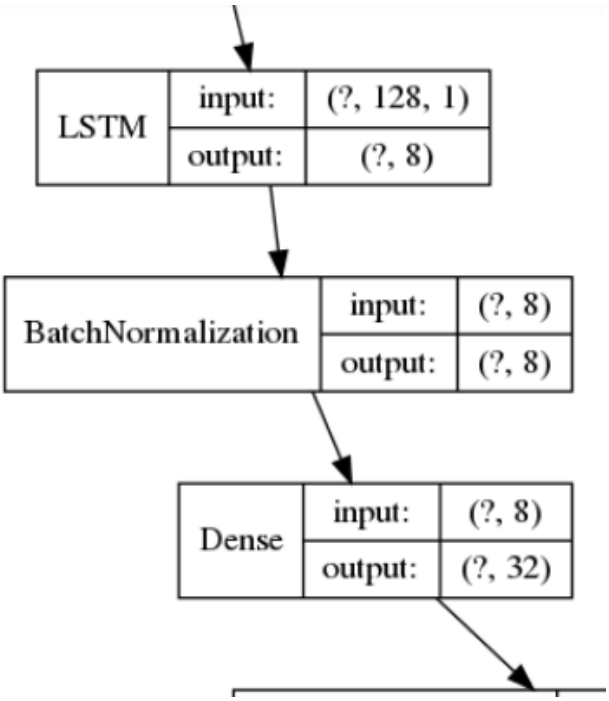

Fig. 5. LSTM encoder-decoder

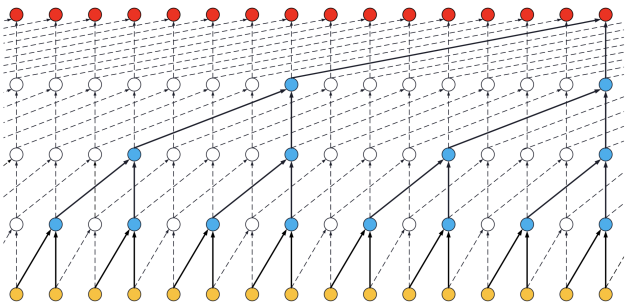

Fig. 6. A stack of 4 dilatated causal convolution layers

of x is defined as follows:

$$(x *_d f)(t) = \sum_{t=0}^{k-1} f(i) \cdot x_{t-d \cdot i}$$

Note how $t - d \cdot i$ enforces the *causal* part of the convolution, as it can only point to sequence elements which are in the past. The dilatation factor d usually increases the further we get into the network, often taking the values of increasing powers of two (i.e. $d = 2^i$ in the i-th hidden layer of the network). This ensures two things. First, that the receptive filed of the network grows exponentially with the number of layers (whereas in an undilated CNN this would grow linearly) granting a longer history. Second, that the output y_t at time t is effectively a function of a contiguous number of steps in the input. A network with m layers has a receptive field of size $2^{m-1}k$.
The used encoder-decoder with TCN has the same structure of the LSTM one but with a TCN layer instead.

7 Attention

As said in section 5, LSTM compress information. However, such compression can weak the long-distance relation patterns to some extent and may fail to highlight important information from historical data. Using attention layers (Vaswani et al. 2017) [2] overcome this problem. Unlike previous seq2seq models, attention models doesn't process data sequentially but process all the sequence together. Attention layers learn temporal and spatial dependencies from the sequence. By processing the all the sequence elements simultaneously model that use attention are highly parallelizable and can take advantage of computational power of GPU outclassing on performance LSTM. An attention function can be described as mapping a query and a set of key-value pairs to an output where the query, keys, values and output are all vectors. The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key. In the *scaled dot-product attention* the input consists of queries and keys of dimensions d_k, and values of dimension d_v. The dot products of the query with all keys is computed, then is divided by $\sqrt{d_k}$, and then a softmax function is applied to obtain the weights on the values. In practice, the attention function is computed on a set of queries simultaneously, packed together into a matrix Q. The keys and values are also packed together into matrices K and V. The matrix of outputs is computed as:

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

The original author that introduced Transformers, one of the most popular model that use attention, also proposed multiheading. In multiheading different attention layer are initialized in random way, then the results is joined. Multiheading has been experimented for river level prediction but it doesn't influence the results, good results have been obtained with only one head, but multiheading increase linearly the computation time.

Model	**Max 16hr prediction error**	
	Reno dataset	**Enza dataset**
Feed-Forward	39.3 cm	**11,1 cm**
TCN	38.4 cm	14.0 cm
Attention model	36.9 cm	13,0 cm
LSTM	**36.8 cm**	11.2 cm

Table 1. Comparison of models results

8 Training

The dataset dimensions are 13 years for the Reno river and 4 years for the Enza river.
Every model has been trained with a different number of epochs since it has been used EarlyStopping. The training was stopped automatically when the loss were not improving anymore, by starting to increase or by decreasing only of a small delta. LSTM had very fast convergence but each single epoch was taking longer compared than other models. Batch size has been set to 32. Adam optimizer has been chosen. SGD has been tried but was resulting in worse results. Dropout has been used to avoid overfitting [13] and has been set to 20%. The dataset has been split in 90% for training and 10% for evaluation.

9 Results

Table 1 is a summary and comparison of the models. The metric evaluated is the model prediction error after 16hr. First we can see that in general the Reno dataset is harder to predict than Enza, even if the model has more data to train with. This could be caused by the fact that evaluation on Enza is done only over 5 months while for Reno is done over 15 months, so more difficult situations to predict are possible. LTSM and Feed-forward models are the ones performing better, even if the other two are not so distant. Indeed, how figure 11 shows all the model capture the overall direction of the river. The difference between them are only about some centimeters.
Figure 11 shows an interesting scenario. River level was going down but a rainfall arrives, that makes the river level increase. We see that all the models predicts an increase but all of them underestimate it. Different parameters and modification has been tried. Increasing the sample length makes training and prediction time longer, and give little advantages after 15 hours past data. This is explainable by the fact that the flow time between rainfall and the river Gauge is 7 hours for Enza dataset and 14 for Reno. Every rainfall

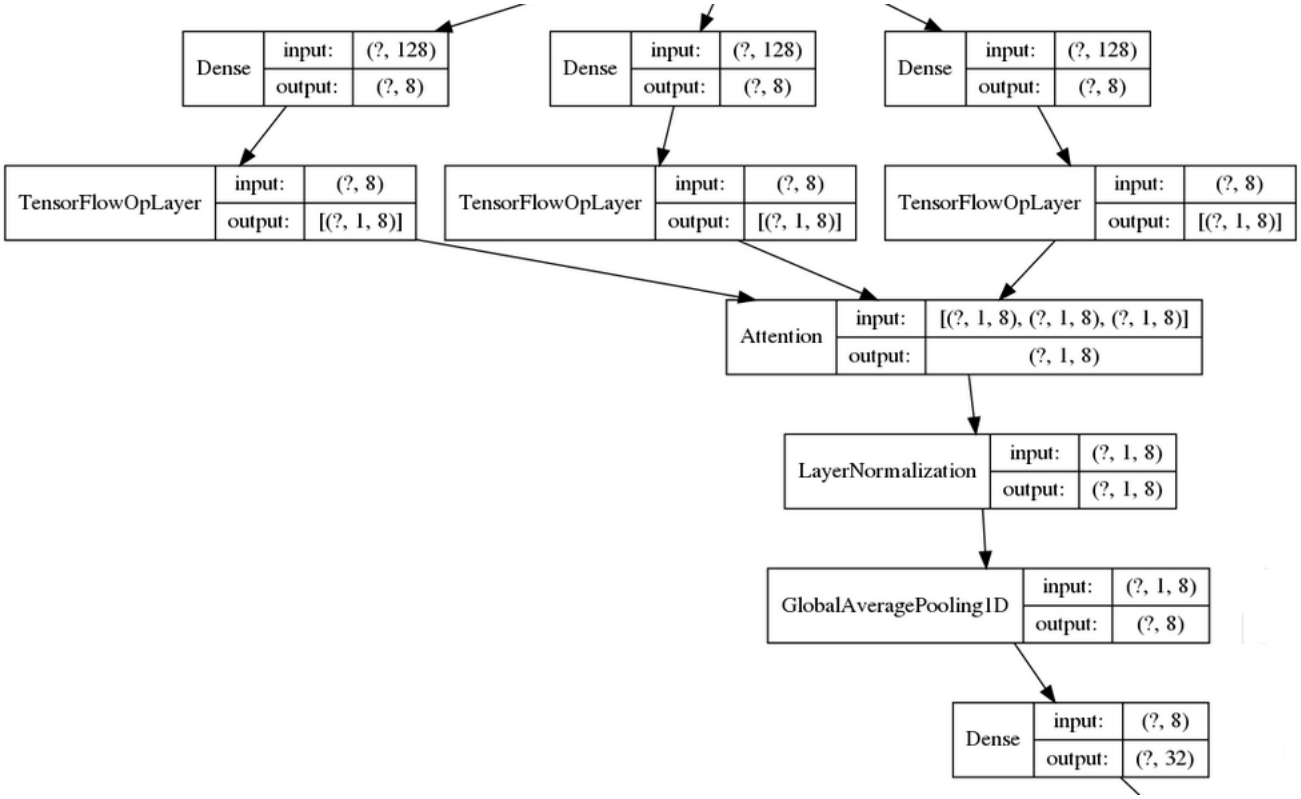

Fig. 7. Encoder-Decoder with attention layer

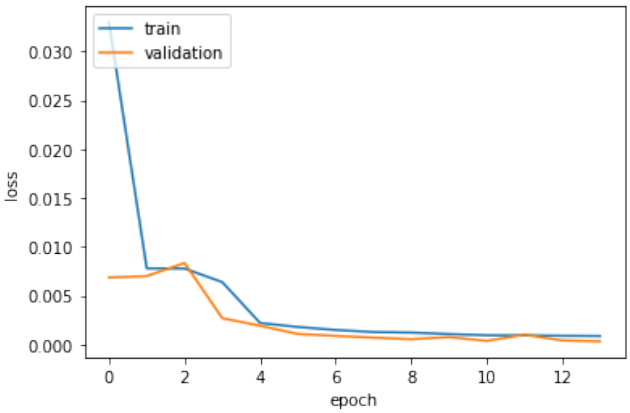

Fig. 8. Feed forward network loss during training

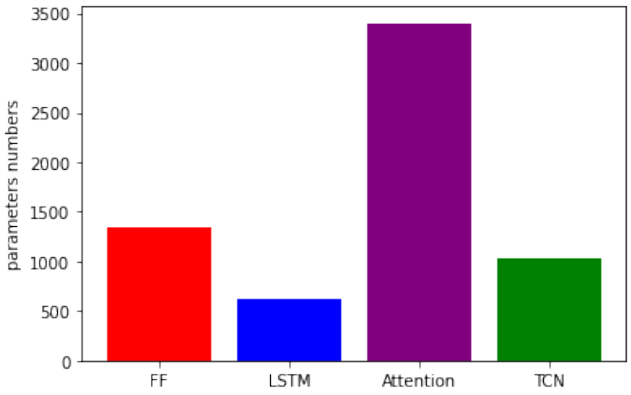

Fig. 9. Encoder-Decoder part parameters number for each model

before 20 hours ago has mostly already escaped away so the current river level is not accountable for that. The increase of computation time related to sample lengths is exponential for LSTM and TCN and linear for the other models. Still, even by taking an LSTM and a TCN with the same number of parameters the difference in time/step is considerable. The error on prediction can mostly be attributed to situation as the one showed in figure 12 where after 3 large rainfall the river level doesn't increase. These situations break our assumptions and creates misbehavior in the network, thus increasing error because the models tends to be more conservative.

10 Conclusion

Four models have been analyzed for the prediction of the river level on 2 different datasets. The problem, unlike what was expected in the beginning, turned out to be at the same time both simple and difficult from different aspects. Simple because even basic models like a Feed-Forward with a few Dense layers can provide an accurate prediction, difficult because to get a better prediction and take more advantage of the ability to extrapolate complex features given by the most recent models, more data would be needed or data consistent with the assumptions made initially. All the models provide good approximations of the general trend of the watercourse.

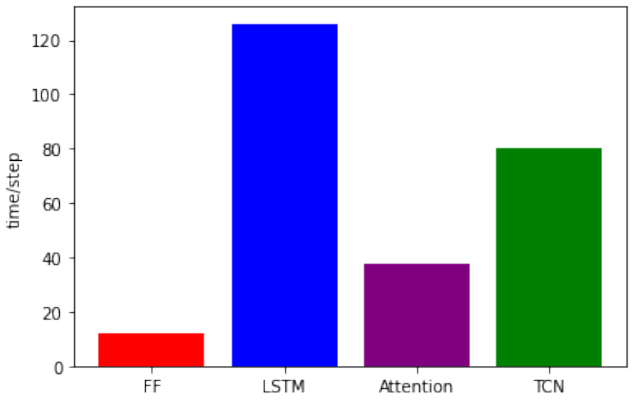

Fig. 10. Computation time for each step for different models

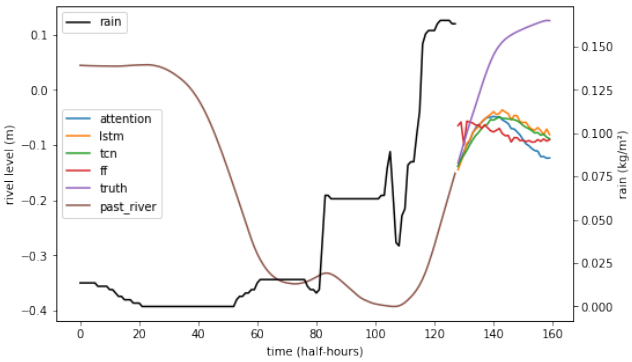

Fig. 11. Example of difference between models predictions

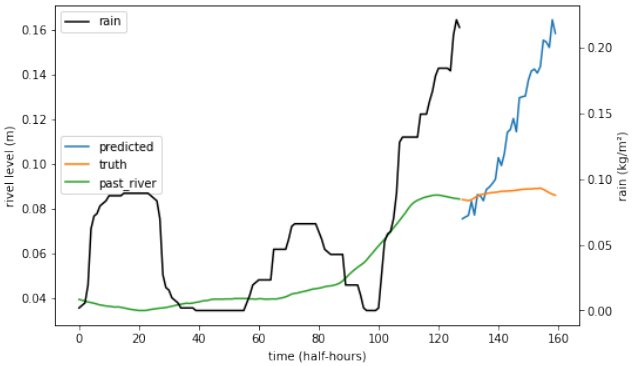

Fig. 12. A situation were the river should have raised but it didn't

The use of a model that approximates the hydrological process has helped to obtain greater accuracy instead of letting the network learn the relationship between the input feature.

11 Future works

Adding rainfall data from more stations upstream could improve prediction since it would increase the network knowledge of actually how much water mass the river is discharging. For larger basins this can be fundamental. Also adding temperature data would help to determine better the natural evaporation of the river, or the dryness level of the ground. The attention model was using a dense layer as an encoder, but it could be interesting to add an attention layer to a LSTM or TCN network. Lastly a greater degree of hydrologic science could be inserted in the model, increasing the bias by adding constraints to the output or by predicting parameters of a well-known hydrological model instead of searching for one.

References

[1] Flood Prediction Using Machine Learning Models: Literature Review, Amir Mosavi, Pinar Ozturk and Kwok-wing Chau, 2018

[2] Attention Is All You Need, Ashish Vaswani and Noam Shazeer and Niki Parmar and Jakob Uszkoreit and Llion Jones and Aidan N. Gomez and Lukasz Kaiser and Illia Polosukhin, 2017

[3] Talos: Hyperparameter Optimization for TensorFlow, Keras and PyTorch.

[4] Dext3r data extraction webapp for Arpae Simc (Servizio Idro-meteo-clima)

[5] Artificial Neural Networks in Hydrology, ASCE Tasck Commitee On Application of Artificial Neural Networks in Hydrology, 2000.

[6] Smith, J., and Eli, R. N. (1995). "Neural-network models of rainfall- runoff process." J. Water Resour. Plng. and Mgmt., ASCE, 121(6), 499–508.

[7] Chen, Xi, et al. "The importance of short lag-time in the runoff forecasting model based on long short-term memory." Journal of Hydrology 589 (2020): 125359.

[8] Yukai Ding, Yuelong Zhu, Jun Feng, Pengcheng Zhang, Zirun Cheng, Interpretable spatio-temporal attention LSTM model for flood forecasting, Neurocomputing, Volume 403, 2020, Pages 348-359

[9] Fernando Nogueira, A Python implementation of global optimization with gaussian processes. Github link.

[10] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780

[11] Bengio, Y., Simard, P., and Frasconi, P.: Learning long-term depen- dencies with gradient descent is difficult, IEEE T. Neural Net- wor., 5, 157–166, 1994.

[12] F. Kratzert1, , D. Klotz , C. Brenner , K. Schulz , and M. Herrnegger: Rainfall–runoff modelling using Long Short-Term Memory (LSTM) network, Hydrol. Earth Syst. Sci., 22, 6005–6022, 2018

[13] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R.: Dropout: A Simple Way to Prevent Neural Networks from Overfitting, J. Mach. Learn. Res., 15, 1929–1958, 2014.

[14] Keras: the Python deep learning API

[15] Sergey Ioffe, Christian Szegedy: Batch Normalization: Accelerating Deep Network training by reducing Internal Covariate Shift

[16] Lea, Colin, et al. :Temporal convolutional networks for action segmentation and detection, 2016.