



CleverRiver: an open source and free Google Colab toolkit for deep-learning river-flow models

Marco Luppichini^{1,2} · Monica Bini^{1,3,4} · Roberto Giannecchini^{1,4,5}

Received: 4 October 2022 / Accepted: 15 November 2022
© The Author(s) 2022, corrected publication 2022

Abstract

In a period in which climate change is significantly varying rainfall regimes and their intensity all over the world, river-flow prediction is a major concern of geosciences. In recent years there has been an increase in the use of deep-learning models for river-flow prediction. However, in this field we can observe two main issues: i) many case studies use similar (or the same) strategies without sharing the codes, and ii) the application of these techniques requires good computer knowledge. This work proposes to employ a Google Colab notebook called CleverRiver, which allows the application of deep-learning for river-flow predictions. CleverRiver is a dynamic software that can be upgraded and modified not only by the authors but also by the users. The main advantages of CleverRiver are the following: the software is not limited by the client hardware, operating systems, etc.; the code is open-source; the toolkit is integrated with user-friendly interfaces; updated releases with new architectures, data management, and model parameters will be progressively uploaded. The software consists of three sections: the first one enables to train the models by means of some architectures, parameters, and data; the second section allows to create predictions by using the trained models; the third section allows to send feedback and to share experiences with the authors, providing a flux of precious information able to improve scientific research.

Keywords CleverRiver · Google Colab · River flow · Forecasting · Deep-learning · Software · Geosciences

Introduction

River flow prediction is an important tool for early flood warning, water resource management, water demand assessment, irrigation, agriculture, and hydroelectric power generation. These aspects become more and more critical in the

case of climate changes causing a variation in rainfall regime and land use in many areas (Merz et al. 2014; Deitch et al. 2017; Blöschl et al. 2019). In particular, extreme weather events produce flash floods, floods, and debris flow phenomena. These have relevant socio-economic implications and represent a significant scientific issue, as confirmed by the extensive literature on the subject (Bates et al. 2008a, b, 2012; Gaume et al. 2016; Bryndal et al. 2017; IPCC 2018).

In recent years, we have observed an increase in the use of deep-learning in geosciences and in particular in river-flow prediction, with promising results (e.g., Boulmaiz et al. 2020; Chattopadhyay et al. 2020; Kratzert et al. 2018; Luppichini et al. 2022; Sit et al. 2020). The implementation of suitable run-off models is made difficult by the complexity of the natural systems and by the environmental information available (Jaiswal et al. 2020). Furthermore, each physically-based model is limited by the inevitable simplifications of the modeled system (Antonetti and Zappa 2018). The deep-learning models available make it possible to manage complex systems without having to introduce any simplifications, information is instead directly extracted from the data. These procedures are the most appropriate for addressing

Communicated by: H. Babaie

✉ Marco Luppichini
marco.luppichini@unifi.it

¹ Department of Earth Sciences, University of Pisa, Via S. Maria, 52, 56126 Pisa, Italy

² Department of Earth Sciences, University of Study of Florence, Via La Pira 4, Florence, Italy

³ Istituto Nazionale di Geofisica e Vulcanologia (INGV), Via Vigna Murata 605, 00143 Rome, Italy

⁴ CIRSEC Centro Interdipartimentale di Ricerca per lo Studio degli Effetti del Cambiamento Climatico dell'Università di Pisa, Via del Borghetto 80, 56124 Pisa, Italy

⁵ Institute of Geosciences and Earth Resources, IGG-CNR, Via Moruzzi 1, 56124 Pisa, Italy

the noisy and chaotic nature of the time-series forecasting problems (Livieris et al. 2020).

Long short-term memory (LSTM) and convolutional neural networks (CNNs) are the most common and most efficient deep-learning methods (Zheng et al. 2019; Yi et al. 2019; Fawaz et al. 2020; Sit et al. 2020). The combination of CNN and LSTM models (CNN-LSTM) allows to exploit the advantages of two different layers. LSTM efficiently acquires sequence pattern information thanks to its peculiar architecture, whereas CNN layers filter the noise in the input data and extract the most significant features for the final prediction model (Bengio et al. 2013). On the other hand, LSTM exploits only the features present in the training set, although they can be adapted to cope with temporal correlations (Livieris et al. 2020). Several works have used deep-learning models based on LSTM networks to create run-off simulations (Kratzert et al. 2018; Le et al. 2019; Boulmaiz et al. 2020; Liu et al. 2020; Nguyen and Bae 2020; Hu et al. 2020), whereas others are based on CNN (Li et al. 2018; Huang et al. 2020; Kim and Song 2020; Hussain et al. 2020), or on a combination of both (CNN-LSTM) (Kimura et al. 2019; Baek et al. 2020; Xu et al. 2020). Other LSTM techniques (LSTM-ED) consider two blocks of layers: the first block (called encoder) reads the input sequence and encodes it into a fixed-length vector, whereas the second block (called decoder) decodes the fixed-length vector and transmits the intended sequence (Sutskever et al. 2014; Cui et al. 2022; Luppichini et al. 2022).

However, these tools require good computer skills that can limit an application of these techniques outside the research community, for example the technical bodies managing the territory. The availability of software and user-friendly toolkits can improve the application of these techniques in several other cases. If the results derived from these toolkits are inserted into a network, it will be possible to obtain increased knowledge, leading to future developments and improvements in the field. However, Sit et al. 2020 observed that similar techniques had been used worldwide for different studies but, apart from some exceptions, these applications are not open-source and reproducible. This is a noteworthy limit to their distribution and application.

The aim of this work is to exploit a dynamic and valid Google Colab toolkit called CleverRiver for the application of deep-learning models for river-flow prediction. This toolkit makes it possible to build workflows using hardware resources made available by the company and not those of the user's desktop PC (Bisong 2019). The toolkit allows the application of deep-learning models based on different architectures, currently the most used to create models for river flow predictions. In particular, the architectures are based on the researches of Luppichini et al. 2022 and Lupi et al. (2022). Google Colab can be employed by two different

types of users: the first user has poor computer skills, or none at all; the second user is able to understand and to interact with the code of the toolkit. In the first case, the user can apply the method of the tool to her/his data to obtain a result and new computational capacity. In the second case, the user can compare the code of the toolkit with her/his own code, and can also contribute to improving the toolkit by bringing new and clear knowledge to the scientific community.

Materials and methods

CleverRiver is projected in close relationship to the work of Luppichini et al. 2022, making it possible to apply their method. The workflow is based on the use of the API of Keras, Tensorflow libraries for the creation of the deep-learning models. The toolkit uses also the Numpy and Pandas libraries for the management of the data. CleverRiver is composed of three sections. The first one aims to train deep-learning models by using a progressive and user-friendly procedure. In this section the user can exploit different types of data (e.g., hydrometric height, discharge, rainfall, temperature) with different data frequency (e.g., daily, hourly) setting the inputs and outputs of the models in simple manner. A deep-learning model can be interpreted as a mathematical expression:

$$\hat{O} = f(X_t) = f((I_{t-1,1}, I_{t-2,1}, \dots, I_{t-n,1}), (I_{t-1,2}, I_{t-2,2}, \dots, I_{t-n,2}), \dots, (I_{t-1,m}, I_{t-2,m}, \dots, I_{t-n,m}))$$

where \hat{O} is the predicted output (hydrometric height or discharge) at time t , and I are the antecedent inputs (e.g., m can be 1 = rainfall, 2 = discharge, 3 = temperature). The choice of the dimension of n depends on the characteristics of data such as sample frequency (daily, hourly, etc.), and on the characteristics of the simulated watershed (e.g., run-off time). These parameters must be chosen by the user after some tests have been performed. For example, Luppichini et al. 2022 set n to 96, using rainfall data with 15 min of frequency corresponding to a maximum antecedent t of 24 h. The authors then simulated the watershed characterized by a fast run-off (in several cases lower than 12 h). By setting these parameters and after uploading the CSV files (e.g., rainfall data) in the workspace, the procedure allows to create the input matrix that will be used to train the models.

For training of the models, the dataset has to be divided into three parts: training, validation, and dataset testing. The training and validation datasets are used during the training steps, whereas the test dataset is used during the evaluation of the results. Dividing the dataset allows the user to reduce the possibility of overfitting. The partition 60%–20%–20% for training, validation and test datasets, respectively, has been used in several studies (Li et al. 2020; Nguyen and

Bae 2020; Hu et al. 2020; Luppichini et al. 2022) and has permitted to dispose of sufficient data for both the training and the evaluation of the model. To train the model, the software allows to select between loss function and optimizer from a list of the most commonly used of the two parameters. CleverRiver provides the use of three different model architectures: i) LSTM; ii) LSTM-ED; iii) CNN-LSTM. These are the most common architectures used for flood prediction (Sit et al. 2020; Cui et al. 2022; Luppichini et al. 2022). The first one is the most straightforward architecture composed of a simple LSTM node and a Dense node. The

LSTM-ED architecture was proposed by Luppichini et al. (2022) and is based on two blocks of LSTM nodes. Finally, the CNN-LSTM architecture proposed by Lupi et al. (2022) is composed of a combination of CNN and LSTM nodes (Fig. 1). The parameters of the architecture size (e.g., number of nodes) can be modified by the user allowing to test different settings.

The last parameters for simulation allow to define the range of the time interval of the predictions. For example, if the dataset has a daily frequency, we can define that the max value of the range is $t = 10$ days and we

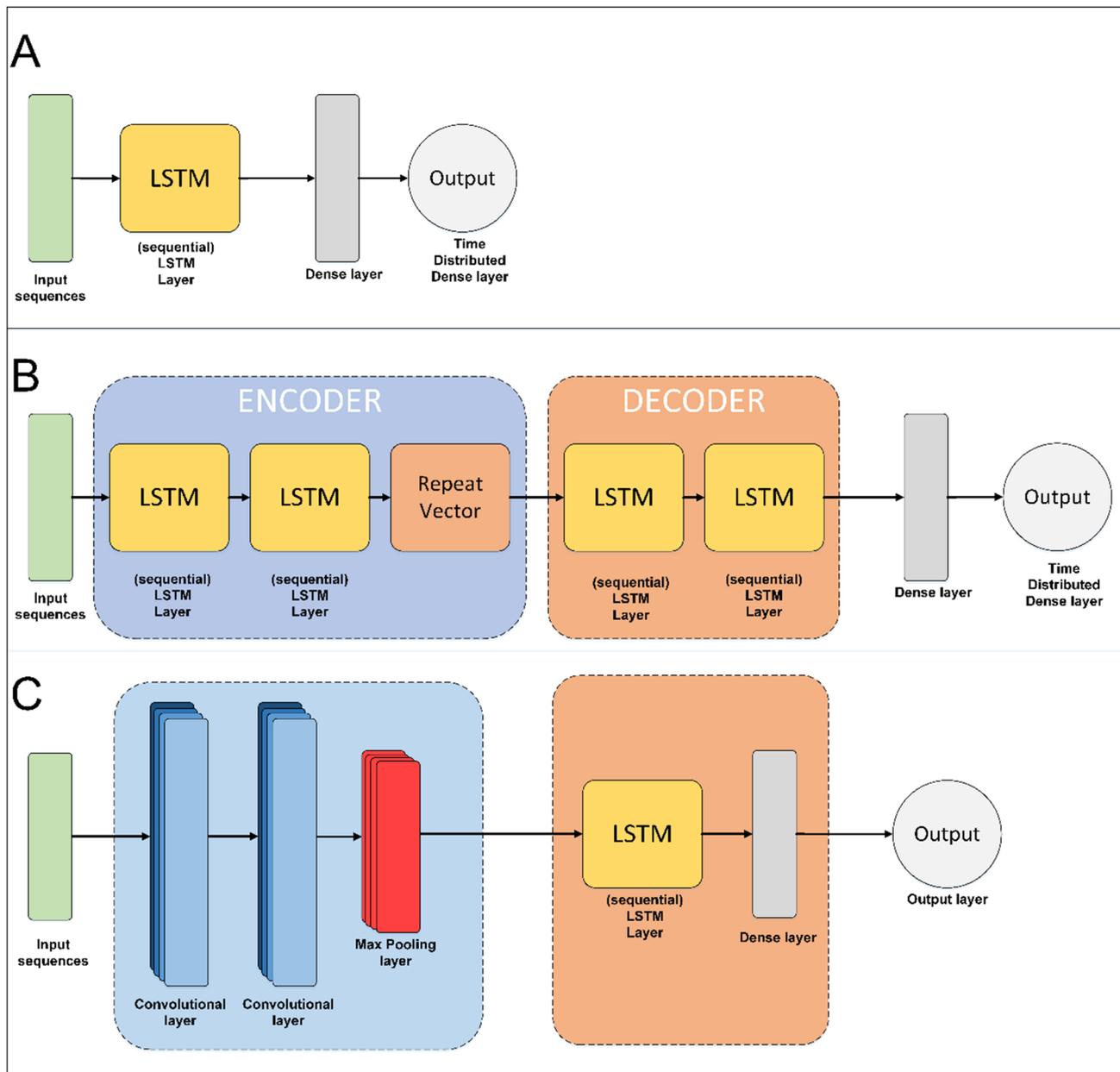
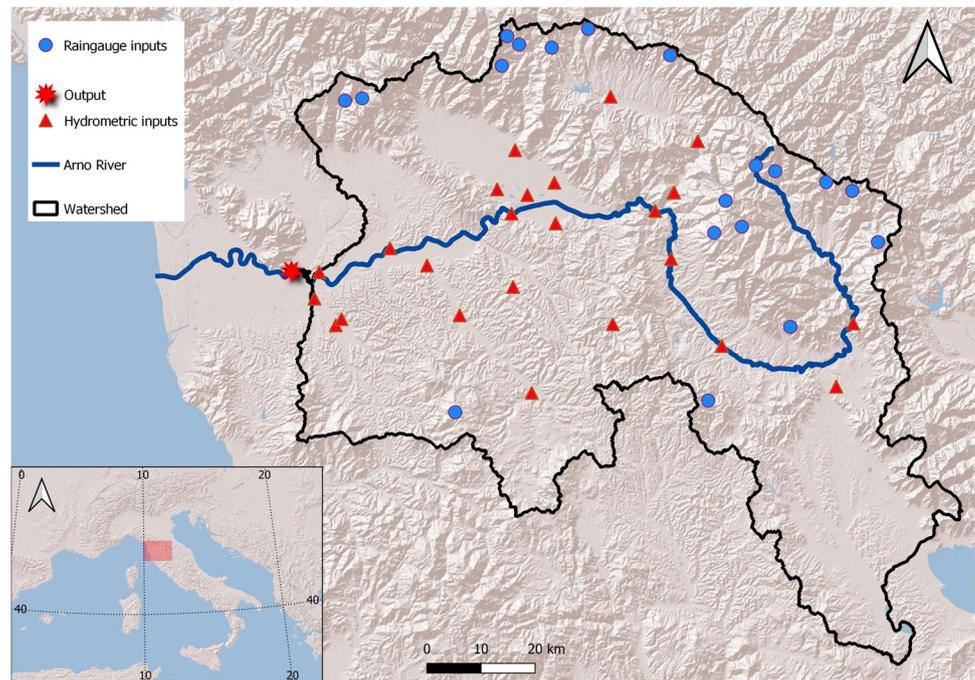


Fig. 1 Deep-Learning architectures developed in CleverRiver: **A** Long Short-Term Memory (LSTM) architecture; **B** encoder-decoder LSTM architecture; **C** Convolutional Neural Network (CNN) and LSTM combined (CNN-LSTM) architectures

Fig. 2 Study area and location of the stations used in this work



can create a simulation each day (step = 1) or every five days (step = 5). The algorithm trains a model for each t of prediction. The following step is model training. During this phase, some graphs and CSV files are compiled, which help to understand the errors of the models. To stop the training, we used the specific API of Keras and specifically the early stopping method. This method allows the training procedure to stop when the monitored metric, namely the value of the cost function, has ceased to improve. Therefore, given all the possible hypotheses, we wanted to find the best one (called “optimal”). This

hypothesis would allow us to make more accurate estimates, still based on the data available.

The first section ends with the possibility to create a unique ZIP file of the output of the models and to transfer it onto the local device.

The second section uses the trained models processed during the first section to create specific simulations of events designed to test and apply the models. In this section, the user can define the time interval of the events and steps between each simulation. In other words, one can choose an interval time (from 2020-01-10 to 2020-01-15) and then decide the

Fig. 3 Steps 1.1 and 1.2: creation of the workspace and importing of the data input

- Section I

▼ Step 1.1: Set the workspace

creation of the **workspace** and the **directory** and import of the needed **libraries**

✓ [1] Execute Step 1.1

[Mostra codice](#)

Ok all configurated!!

▼ Step 1.2: Import the data

Import manually the input data in CSV format into the folder "input_data". If you do not see the folder, use the reload button. The input files must have two columns: 'date' and 'value'. Each csv must have a time series of a station. The CSV files of the model output have to be inserted in the "training_input_data" folder and called "output.csv". The column separator must be the comma and the decimal separator the dot. Then press the run button to check the imported data.

✓ [2] Execute Step 1.2

[Mostra codice](#)

All files have been loaded correctly

Step 1.3: Set parameters for the matrix input

Define the number of **back shift** (variable n_step_back) and the **frequency** of step to give (variable n_step). For example for 15 minutes frequency time series, we suggest **96 step back** (24 h x 4 times) with a **step of 4** (one each hour).

For more information see the equation 1 in Luppichini et al. 2022:

$$H = f(X_t) = f(H_{(t-1)}, H_{(t-2)}, H_{(t-n)}, \dots, R_{(t-1)}, R_{(t-2)}, \dots, H_{(t-n)})$$

where H stands for the predicted hydrometric height at time t; $H_{(t-1)}, H_{(t-2)}, \dots, H_{(t-n)}$ are the antecedent hydrometric heights (up to t-1, t-2, ..., t-n time steps); $R_{(t-1)}, R_{(t-2)}, \dots, R_{(t-m)}$ are the antecedent rainfall (t-1, t-2, ..., t-m time steps).

Some tests and trials highlighted the need to create an input dataset with t up to 96 steps. To decrease the noise contained by many steps and close measurements, we provided every t for the first previous hour and then one every 4 steps (e.g., t-0, t-1, t-2, t-3, t-4, t-8, t-12, t-16, ..., t-96) up to the 24th hour.

```
[7] Set parameters
n_step_back: 24
n_step: 1
Mostra codice
```

Step 1.4: Create the input matrix

Create the input matrix to train the model

```
It may take a few minutes
Mostra codice

/usr/local/lib/python3.7/dist-packages/spykernel_launcher.py:56: PerformanceWarning: DataFrame is highly fragmented. This is usually the result of calling 'frame.insert' many times, which has poor performance. Consider j
/usr/local/lib/python3.7/dist-packages/spykernel_launcher.py:53: PerformanceWarning: DataFrame is highly fragmented. This is usually the result of calling 'frame.insert' many times, which has poor performance. Consider j
Information input matrix:
Length of time series: 63530
Records eliminated for missing values: 32206
N columns 1059
N data in matrix: 67214740
```

Fig. 4 Steps 1.3 and 1.4: definition of the parameters and creation of the input dataset

beginning of each simulation (e.g., each day, every two days), until the interval is complete. The results can be exported using the functionalities of Google Colab, or running a specific step that creates a ZIP of the work environment.

The third section aims to create a relationship between the users and the CleverRiver authors, with a form that allows to send a message directly to the authors, inviting the users to share their toolkits and experiences.

CleverRiver is loaded onto a GitHub repository together with the documentation and datasets having different characteristics (e.g., number of stations, sample frequency) for experimentation (<https://github.com/mluppichini/CleverRiver>). In this work, the CleverRiver results are derived using the “dataset2” uploaded onto the GitHub repository, composed of 25 hydrometric height time series and 19 rainfall time series of the Arno River

```
Set train, validation and test dataset sizes as integer percentage values (e.g, 60)
train_size: 60
val_size: 20
test_size: 20

Model parameters
If the select model does not include CNN the number of CNN nodes will be set to 0
model_select: LSTM-ED
cnn_n_node: 8
lstm_n_node: 32
patience: 100
optimizer: Adam
loss_function: MSE

Chose the forward prediction maximum t and the step between each prediction.
max_t_forecasting: 24
step_t: 1
Show code
```

Fig. 5 Step 1.5: definition of the parameters for model training

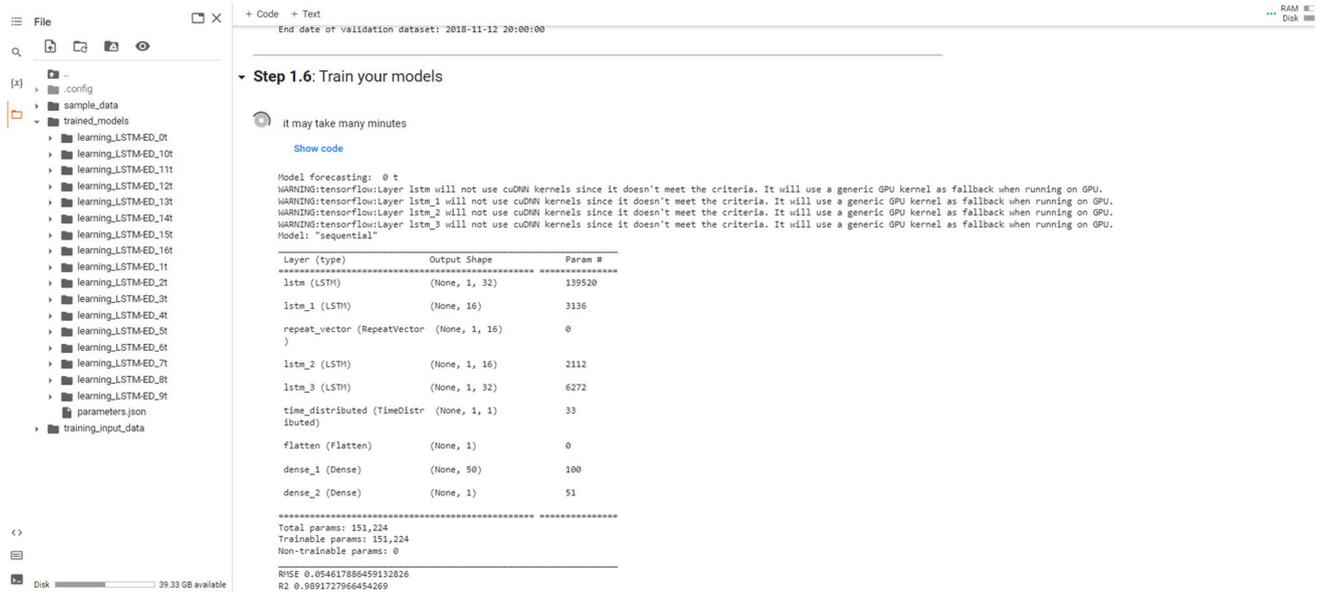


Fig. 6 Step 1.5: training of models

Fig. 7 History graph of the training procedure. The graph is referred to a simulation of 0 h of forecast (t_0)

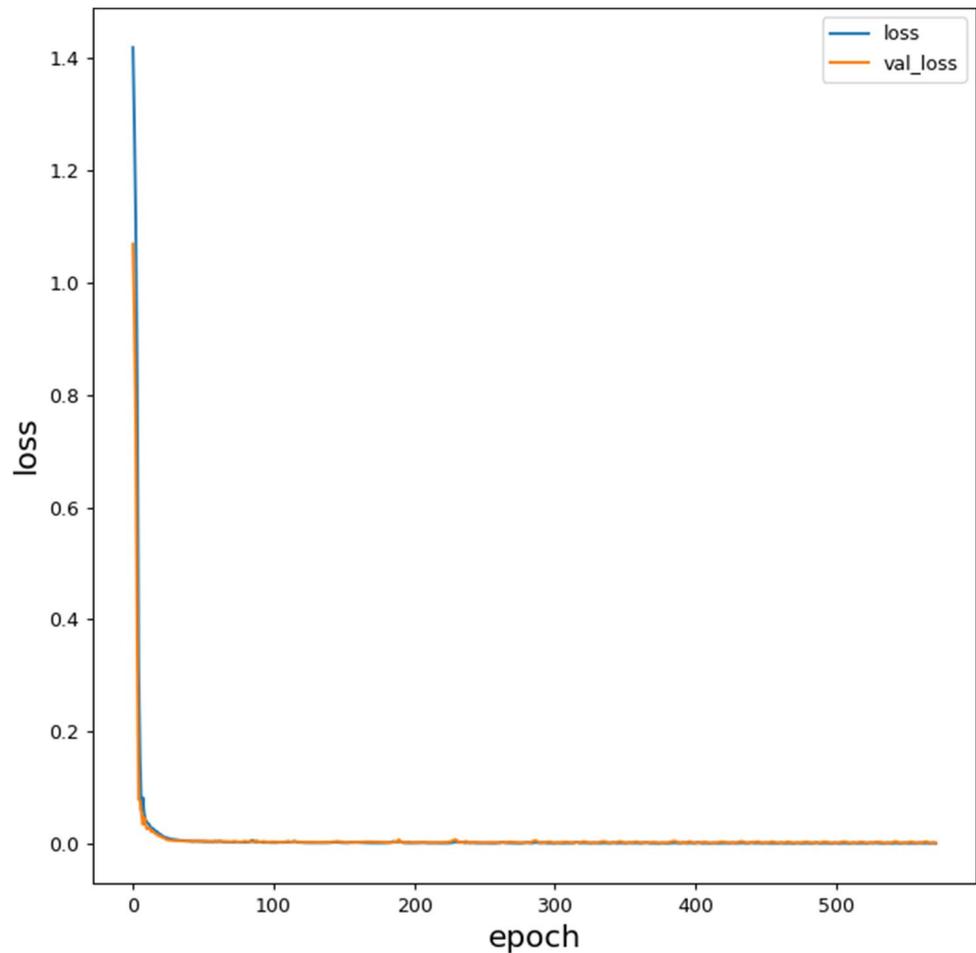


Fig. 8 Scatter plot of the observed and predicted values. The graph is referred to a simulation of 0 h of forecast (t_0)

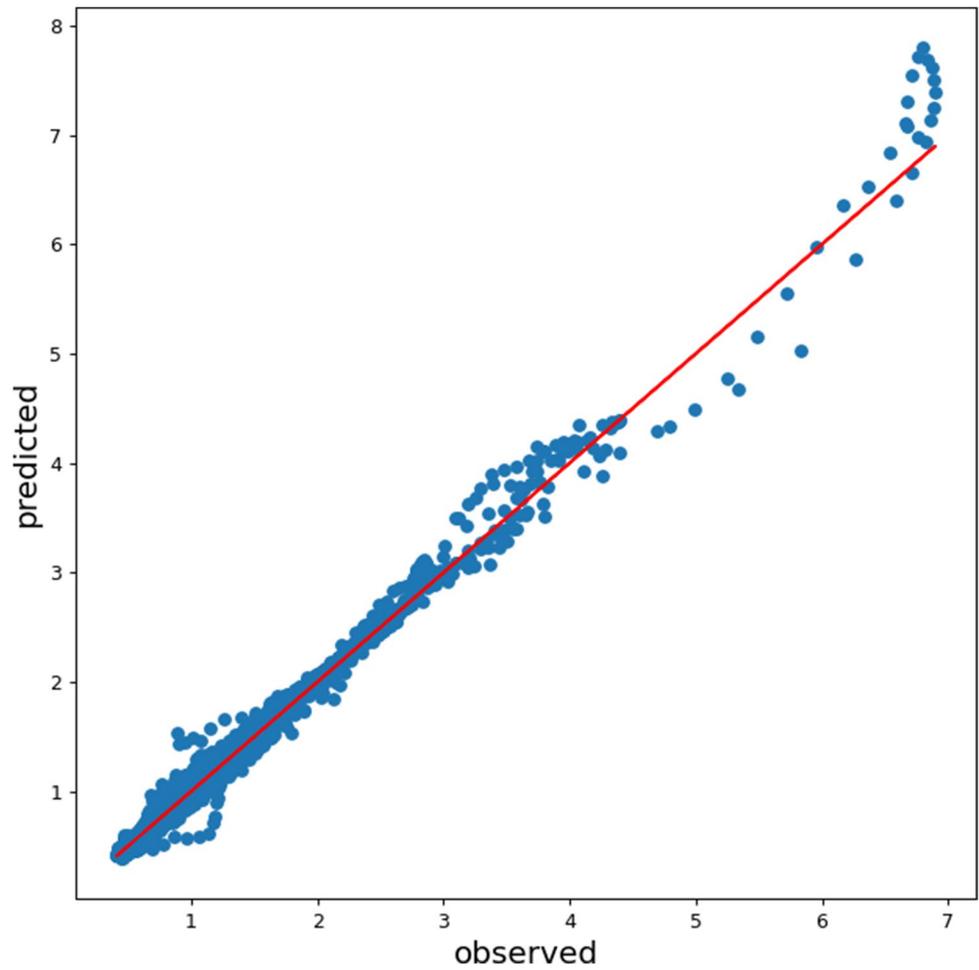
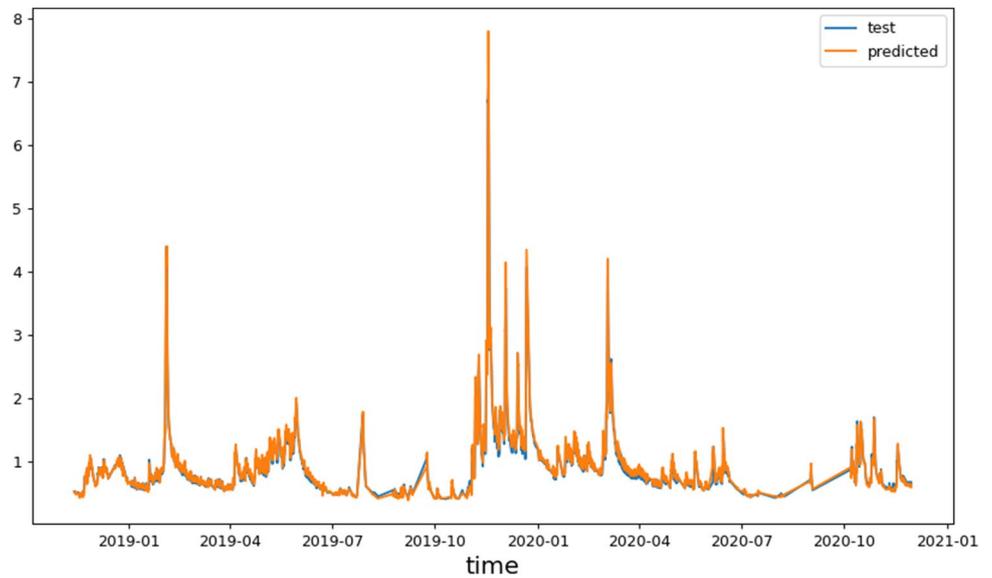


Fig. 9 Plot of the observed and predicted values of the test dataset. The graph is referred to a simulation of 0 h of forecast (t_0)



(Tuscany, Italy) watershed. The data has an accumulation period of 1 h from 2010 to 2020. This dataset is used and analyzed in more detail by Luppichini et al. 2022, who applied the same methodologies for the analysis of

the precision and accuracy of the deep learning models on this study area. Figure 2 shows the location of the stations composing the dataset used to describe the workflow.

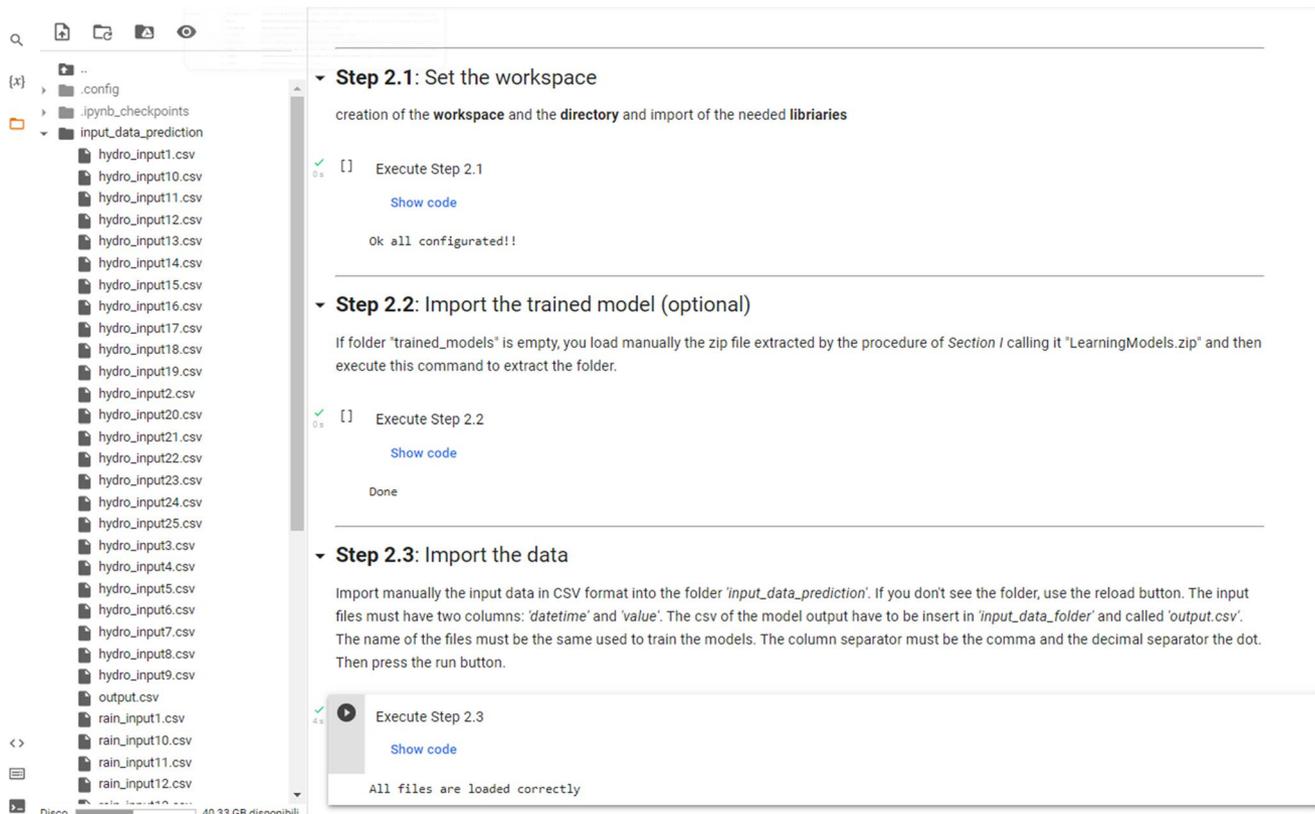


Fig. 10 Steps 2.1, 2.2 and 2.3: creation of the workspace and import of trained models and data input for simulations

Software description

CleverRiver is installed by importing the necessary libraries and setting up of the workspace (Step 1.1 in Fig. 3). The tool then prompts the user to import the input data in the “*training_input_data*” directory; the notebook checks whether the files are correct for the following procedures (Step 1.2 in Fig. 3). CleverRiver can create the input matrix through Steps 1.3 and 1.4. The algorithm provides some information on the size of the input matrix, such as number of records, number of columns, number of data (Fig. 4). The user can define the model parameters with Step 1.5 by using a simple user-friendly interface (Fig. 5). Step 1.6 allows to train the models. For each simulation, CleverRiver provides the structure of the model and the errors calculated on the test dataset of the best model expressed in terms of Root Mean Square Error (RMSE) and Mean Absolute Error (MAE). The outputs are saved in the “*trained_models*” directory and each training has its own directory. For each simulation, the algorithm saves the model in JSON and H5 format. It also saves a CSV file including the information and the errors of the model; the predicted values calculated on the test dataset in CSV format; and three graphs (Fig. 6).

The first graph is a plot of the training history of the loss function value calculated on the training and on the validation dataset (Fig. 7). The second graph is a scatter plot showing the relationship between observed and predicted values (Fig. 8). The third graph is a time plotter of the test dataset with observed and predicted values (Fig. 9).

Section [Materials and methods](#) allows to use the trained model to create specific simulations. With Steps 2.1, 2.2 and 2.3, CleverRiver creates the workspace and imports the trained models and data for the simulations. With Step 2.4, it creates the input matrix as for Step 1.4, and it saves the predicted values in the “*output_predictions*” directory.

It is then possible to execute Step 2.5 for the simulation of a specific event. In this step, the algorithm prompts some simple inputs: i) the time interval to simulate; ii) the time distance between each simulation; iii) the label of the y axis for the output graphs (Fig. 10). For each simulated event, the plots (Fig. 11) are saved in the output folder and can be downloaded using the Google Colab functions or Step 2.6, which allows to create a ZIP file of the “*output_predictions*” directory.

Table 1 summarizes the parameters required by the toolkit with a brief description, the value ranges, and the default values.

Fig. 11 Step 2.5: setting the simulation of the event and result of the graph (the coloured curves are the successive simulations with a time distance of 6 h, whereas the black line represents the observed values)

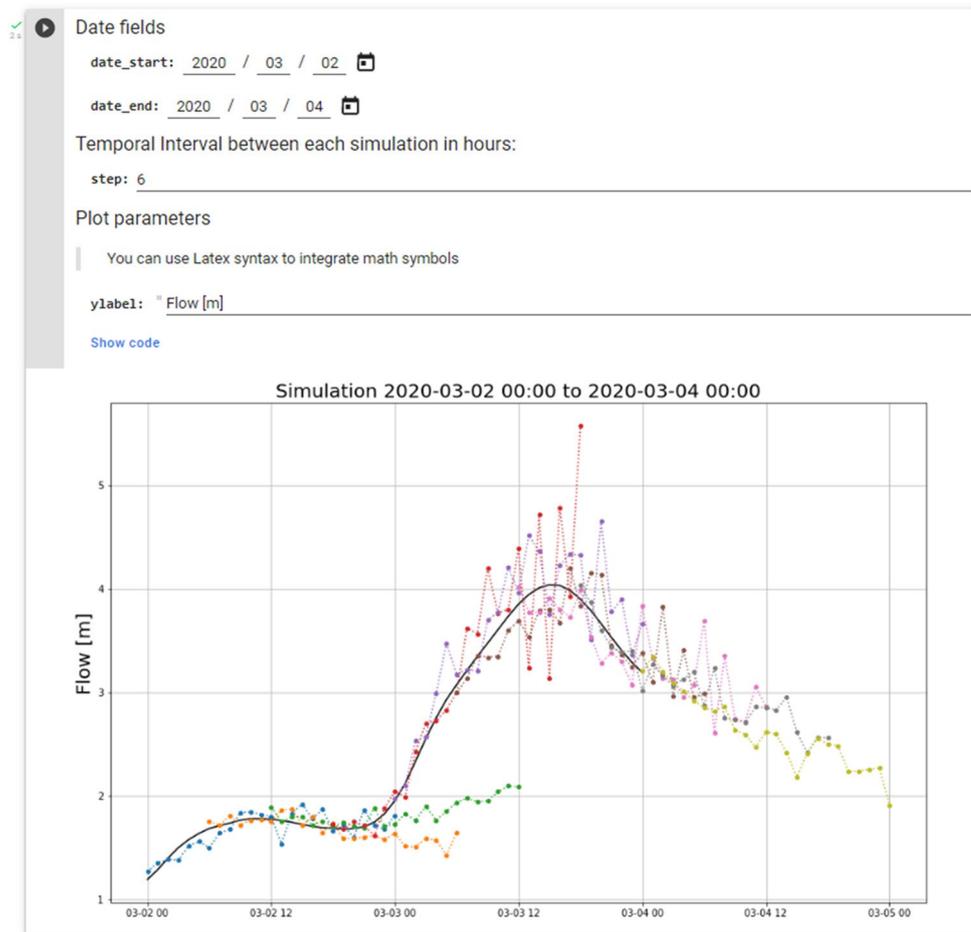


Table 1 Parameters of the toolkit with a brief description, the value ranges and the default values

Section	Step	Parameter	Description	Value Range	Default Value
I	1.3	n_step_back	Number of back shifts	0 - inf	24
I	1.3	n_step	Frequency of the step	1 - inf	1
I	1.5	train_size	Train dataset size as a percentage of the entire dataset	1 - 99	60
I	1.5	val_size	Validation dataset size as a percentage of the entire dataset	1 - 99	20
I	1.5	test_size	Test dataset size as a percentage of the entire dataset	1 - 99	20
I	1.5	model_select	Choice of the model architecture as showed in Fig. 1	LSTM; LSTM-ED; CNN-LSTM	LSTM-ED
I	1.5	cnn_n_node	Number of nodes of the CNN layers	0 - 128	0
I	1.5	lstm_n_node	Number of nodes of the LSTM layers	0 - 128	32
I	1.5	patience	Number of epochs with no improvement after which training will be stopped	10-1000	100
I	1.5	optimizer	Optimizer used to minimize the loss function	Adam; SGD; RMSProp	Adam
I	1.5	loss_function	Loss function	MSE; MAE	MSE
I	1.5	max_t_forecasting	Range of the prediction	-	24
I	1.5	step_t	Dimension of the step for the range of the prediction	1 - inf	1
II	2.5	date_start	Starting date of the simulation	-	-
II	2.5	date_end	Ending date of the simulation	-	-
II	2.5	step	Temporal period between each simulation	1 - inf	6
II	2.5	ylabel	Y-axis label of the output graph	-	-

Section III

We wish to share your experiences.
Send a report with suggestions and observations or share the Colab notebook with marco.luppichini@unifi.it. We will be happy to answer you.

name: "Inserisci text qui"

last_name: "Inserisci text qui"

email: "Inserisci text qui"

affiliation: "Inserisci text qui"

text: "Inserisci text qui"

[Mostra codice](#)

Fig. 12 Section [Software description](#) and the form to share issues, results, or general observations with the CleverRiver authors

Section [Software description](#) is a form that allows to contact the CleverRiver authors, so as to create a network for different applications of river flood prediction (Fig. 12).

Discussion and conclusions

Google Colab notebooks are important tools for creating dynamic workspaces with no limits for the client in terms of operating system, Python installation, and hardware (Bisong 2019; Yang et al. 2022). CleverRiver is the first deep-learning software for the prediction of river-flow, and it provides valid techniques based on the most common approaches (Sit et al. 2020; Van et al. 2020; Luppichini et al. 2022) for training of the models and evaluation of the results. CleverRiver is an open-source Python toolkit for the simulation of river flows and can be a reference point for the dissemination of deep-learning models in this field. This toolkit is based on the LSTM and CNN layers, which are probably the most popular, efficient, and commonly used deep-learning techniques (Fawaz et al. 2020; Yi et al. 2019; Zheng et al. 2019). These types of layers have been used in several works with the purpose of predicting river-flow (e.g., Li et al. 2018; Baek et al. 2020; Boulmaiz et al. 2020; Huang et al. 2020; Kim and Song 2020; Van et al. 2020; Hussain et al. 2020; Luppichini et al. 2022). For this reason, CleverRiver is a valid toolkit able to apply this, or similar architectures, in a potentially large number of future applications.

River-flow models based on deep-learning cannot yet be used on a large scale as they require particular computational skills. This is the main difference from physical models, which use different types of free, for-pay, open-source, and non open-source software.

Importantly, the ability to simulate efficient river-flow with the great number of data available in different parts of the world is crucial for present-day river management and geo-risks. In

this regard, CleverRiver represents a valuable tool for a range of potential users including (but not limited to):

- policy makers responsible for regulating river development;
- river managers and engineers designing and implementing flood protection;
- researchers evaluating the impacts of climate change within the fluvial zone;
- students and neophytes to deep-learning techniques, who will be able to learn and try out their datasets.

Finally, the growing demographic pressure on fluvial zones and the changes caused by climate change in high-frequency and high-intensity precipitation events strongly suggest the need to plan for future adaptation of the community (Bates et al. 2008a, b, 2012; Gaume et al. 2016; Bryndal et al. 2017; IPCC 2018).

New releases will be progressively uploaded with new architectures, data management, and model parameters. For these reasons, we think that CleverRiver can be a valid tool to solve the problem of the scarce availability of open-source codes for flood prediction (Sit et al. 2020) and to extend the use of these tools outside the scientific community by means of a preliminary and cognitive approach.

Acknowledgements We thank the three referees for their suggestions which improved the manuscript.

Author contributions Conceptualization, M.L.; methodology, M.L.; software, M.L.; validation, M.L., M.B. and R.G.; formal analysis, M.L., M.B. and R.G.; investigation, M.L.; resources, M.B. and R.G.; data curation, M.L., M.B. and R.G.; writing—original draft preparation, M.L., M.B. and R.G.; writing—review and editing, M.L., M.B. and R.G.; visualization, M.L., M.B. and R.G.; supervision, M.B. and R.G.; project administration, M.B. and R.G.; funding acquisition, M.B. and R.G. All authors have read and agreed to the published version of the manuscript.

Funding Open access funding provided by Università di Pisa within the CRUI-CARE Agreement. This research was funded by the

collaborative research agreement no. 579999-2019 “Autorità di Bacino Distrettuale Appennino Settentrionale” (Resp. Monica Bini and Roberto Giannecchini) and by the project “Cambiamenti globali e impatti locali: conoscenza e consapevolezza per uno sviluppo sostenibile della pianura Apuo-versiliense” Fondazione Cassa di Risparmio di Lucca (call 2018 years 2019–2022- Resp. M. Bini).

Data availability You can contact Marco Luppichini (marco.luppichini@unifi.it) for data and materials.

Declarations

Competing interests The authors declare no competing interests.

Conflicts of Interest The authors declare no conflict of interest.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article’s Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article’s Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Antonetti M, Zappa M (2018) How can expert knowledge increase the realism of conceptual hydrological models? A case study based on the concept of dominant runoff process in the Swiss Pre-Alps. *Hydrol Earth Syst Sci* 22:4425–4447. <https://doi.org/10.5194/hess-22-4425-2018>
- Baek S-S, Pyo J, Chun JA (2020) Prediction of water level and water quality using a CNN-LSTM combined deep learning approach. *Water* 12:3399
- Bates B, Kundzewicz Z, Wu S et al (2008a) Climate change and water
- Bates B, Kundzewicz ZW, Wu S et al (2008b) Climate change and water. Technical paper of the intergovernmental panel on climate change
- Bates BC, Chandler RE, Bowman AW (2012) Trend estimation and change point detection in individual climatic series using flexible regression methods. *J Geophys Res Atmos* 117:D16106. <https://doi.org/10.1029/2011JD017077>
- Bengio Y, Courville A, Vincent P (2013) Representation learning: a review and new perspectives. *IEEE Trans Pattern Anal Mach Intell* 35:1798–1828. <https://doi.org/10.1109/TPAMI.2013.50>
- Bisong E (2019) Google colab. In: Bisong E (ed) Building machine learning and deep learning models on google cloud platform: a comprehensive guide for beginners. Apress, Berkeley, pp 59–64
- Blöschl G, Hall J, Viglione A et al (2019) Changing climate both increases and decreases European river floods. *Nature* 573:108–111. <https://doi.org/10.1038/s41586-019-1495-6>
- Boulmaiz T, Guermoui M, Boutaghane H (2020) Impact of training data size on the LSTM performances for rainfall–runoff modeling. *Model Earth Syst Environ* 6:2153–2164. <https://doi.org/10.1007/s40808-020-00830-w>
- Bryndal T, Franczak P, Krocak R et al (2017) The impact of extreme rainfall and flash floods on the flood risk management process and geomorphological changes in small Carpathian catchments: a case study of the Kasiniczanka river (Outer Carpathians, Poland). *Nat Hazards* 88:95–120. <https://doi.org/10.1007/s11069-017-2858-7>
- Chattopadhyay A, Nabizadeh E, Hassanzadeh P (2020) Analog forecasting of extreme-causing weather patterns using deep learning. *J Adv Model Earth Syst* 12:e2019MS001958. <https://doi.org/10.1029/2019MS001958>
- Cui Z, Zhou Y, Guo S et al (2022) Effective improvement of multi-step-ahead flood forecasting accuracy through encoder-decoder with an exogenous input structure. *J Hydrol (Amst)* 609:127764. <https://doi.org/10.1016/j.jhydrol.2022.127764>
- Deitch MJ, Sapundjieff MJ, Feirer ST (2017) Characterizing precipitation variability and trends in the world’s mediterranean-climate areas. *Water (Basel)* 9:259. <https://doi.org/10.3390/w9040259>
- Fawaz HI, Forestier G, Weber J et al (2020) Deep learning for time series classification: a review. To cite this version: HAL Id: hal-02365025
- Gaume E, Borga M, Llassat MC et al (2016) Mediterranean extreme floods and flash floods. In: The mediterranean region under climate change. A scientific update. IRD Editions, pp 133–144
- Hu Y, Yan L, Hang T, Feng J (2020) Stream-flow forecasting of small rivers based on LSTM
- Huang C, Zhang J, Cao L et al (2020) Robust forecasting of river-flow based on convolutional neural network. *IEEE Trans Sustain Comput* 5:594–600. <https://doi.org/10.1109/TSUSC.2020.2983097>
- Hussain D, Hussain T, Khan A et al (2020) A deep learning approach for hydrological time-series prediction: a case study of Gilgit river basin. *Earth Sci Inform* 13:1–13. <https://doi.org/10.1007/s12145-020-00477-2>
- IPCC (2018) Global warming of 1.5°C. An IPCC Special Report on the impacts of global warming of 1.5°C above pre-industrial levels and related global greenhouse gas emission pathways, in the context of strengthening the global response to the threat of climate change
- Jaiswal RK, Ali S, Bharti B (2020) Comparative evaluation of conceptual and physical rainfall–runoff models. *Appl Water Sci* 10:48. <https://doi.org/10.1007/s13201-019-1122-6>
- Kim DY, Song CM (2020) Developing a discharge estimation model for ungauged watershed using CNN and hydrological image. *Water (Basel)* 12:3534. <https://doi.org/10.3390/w12123534>
- Kimura N, Yoshinaga I, Sekijima K et al (2019) Convolutional neural network coupled with a transfer-learning approach for time-series flood predictions. *Water (Basel)* 12:96. <https://doi.org/10.3390/w12010096>
- Kratzert F, Klotz D, Brenner C et al (2018) Rainfall – runoff modeling using Long Short-Term Memory (LSTM) networks. *Hydrol Earth Syst Sci* 22:6005–6022
- Le XH, Ho H, Lee G, Jung S (2019) Application of Long Short-Term Memory (LSTM) neural network for flood forecasting. *Water (Basel)* 11:1387. <https://doi.org/10.3390/w11071387>
- Li X, Du Z, Song G (2018) A method of rainfall runoff forecasting based on deep convolution neural networks. In: 2018 Sixth international conference on advanced Cloud and Big Data (CBD), pp 304–310
- Li W, Kiaghadi A, Dawson C (2020) High temporal resolution rainfall–runoff modeling using long-short-term-memory (LSTM) networks. *Neural Comput Appl*. <https://doi.org/10.1007/s00521-020-05010-6>
- Liu D, Jiang W, Mu L, Wang S (2020) Streamflow prediction using deep learning neural network: case study of Yangtze River. *IEEE Access* 8:90069–90086. <https://doi.org/10.1109/ACCESS.2020.2993874>
- Livieris IE, Pintelas E, Pintelas P (2020) A CNN–LSTM model for gold price time-series forecasting. *Neural Comput Appl* 32:17351–17360. <https://doi.org/10.1007/s00521-020-04867-x>

- Lupi A, Luppichini M, Barsanti M, Giannecchini R (2022) Deep learning models to complete rainfall time series databases affected by missing or anomalous data. *Earth Science Informatics*. Submitted
- Luppichini M, Barsanti M, Giannecchini R, Bini M (2022) Deep learning models to predict flood events in fast-flowing watersheds. *Sci Total Environ* 813:151885. <https://doi.org/10.1016/j.scitotenv.2021.151885>
- Merz B, Aerts J, Arnbjerg-Nielsen K et al (2014) Floods and climate: emerging perspectives for flood risk assessment and management. *Nat Hazard* 14:1921–1942. <https://doi.org/10.5194/nhess-14-1921-2014>
- Nguyen DH, Bae D-H (2020) Correcting mean areal precipitation forecasts to improve urban flooding predictions by using long short-term memory network. *J Hydrol (Amst)* 584:124710. <https://doi.org/10.1016/j.jhydrol.2020.124710>
- Sit M, Demiray BZ, Xiang Z et al (2020) A comprehensive review of deep learning applications in hydrology and water resources. *Water Sci Technol*. <https://doi.org/10.2166/wst.2020.369>
- Sutskever I, Vinyals O, Le QV (2014) Sequence to sequence learning with neural networks. In: *Advances in neural information processing systems*
- Van SP, Le HM, Thanh DV et al (2020) Deep learning convolutional neural network in rainfall–runoff modelling. *J Hydroinf* 22:541–561. <https://doi.org/10.2166/hydro.2020.095>
- Xu W, Jiang Y, Zhang X et al (2020) Using long short-term memory networks for river flow prediction. *Hydrol Res* 51:1358–1376. <https://doi.org/10.2166/nh.2020.026>
- Yang J, Liu H, Tang Z et al (2022) Visualization of aqueous geochemical data using python and WQChartPy. *Groundwater*. <https://doi.org/10.1111/gwat.13185>
- Yi A, Li Z, Gan M et al (2019) A deep learning approach on short-term spatiotemporal distribution forecasting of dockless bike-sharing system. *Neural Comput Appl* 31:1–13. <https://doi.org/10.1007/s00521-018-3470-9>
- Zheng J, Fu X, Zhang G (2019) Research on exchange rate forecasting based on deep belief network. *Neural Comput Appl* 31:573–582. <https://doi.org/10.1007/s00521-017-3039-z>

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.