Hydrological Process Surrogate Modelling and Simulation with Neural Networks

Ruixi Zhang[1(✉)], Remmy Zen[1], Jifang Xing[1], Dewa Made Sri Arsa[2], Abhishek Saha[3,4], and Stéphane Bressan[1]

[1] National University of Singapore, 21 Lower Kent Ridge Rd, Singapore, Singapore
{zhangruixi,remmy}@u.nus.edu, jifang@comp.nus.edu.sg, steph@nus.edu.sg
[2] Udayana University, Denpasar, Bali, Indonesia
dewamsa@unud.ac.id
[3] Delft University of Technology, Mekelweg 5, 2628 Delft, Netherlands
a.saha@tudelft.nl
[4] Hydroinformatics Institute, 77 Science Park Drive, Singapore, Singapore
abhishek@h2i.sg

Abstract. Environmental sustainability is a major concern for urban and rural development. Actors and stakeholders need economic, effective and efficient simulations in order to predict and evaluate the impact of development on the environment and the constraints that the environment imposes on development. Numerical simulation models are usually computation expensive and require expert knowledge. We consider the problem of hydrological modelling and simulation. With a training set consisting of pairs of inputs and outputs from an off-the-shelves simulator, We show that a neural network can learn a surrogate model effectively and efficiently and thus can be used as a surrogate simulation model. Moreover, we argue that the neural network model, although trained on some example terrains, is generally capable of simulating terrains of different sizes and spatial characteristics.

Keywords: Surrogate model · Neural networks hydrological · Simulation

1 Introduction

An article in the Nikkei Asian Review dated 13 September 2019 warns that both the cities of Jakarta and Bangkok are sinking fast. These iconic examples are far from being the only human developments under threat. The United Nation Office for Disaster Risk Reduction reports that the lives of millions were affected by the devastating floods in South Asia and that around 1,200 people died in the Bangladesh, India and Nepal [30]. Climate change, increasing population density, weak infrastructure and poor urban planning are the factors that increase the risk of floods and aggravate consequences in those areas. Under such scenarios, urban and rural development stakeholders are increasingly concerned with the interactions between the environment and urban and rural development.

H. W. Lauw et al. (Eds.): PAKDD 2020, LNAI 12085, pp. 449–461, 2020.
https://doi.org/10.1007/978-3-030-47436-2_34

In order to study such complex interactions, stakeholders need effective and efficient simulation tools.

A flood occurs with a significant temporary increase in discharge of a body of water. In the variety of factors leading to floods, heavy rain is one of the prevalent [17]. When heavy rain falls, water overflows from river channels and spills onto the adjacent floodplains [8]. The hydrological process from rainfall to flood is complex [13]. It involves nonlinear, time-varying interactions between rain, topography, soil types and other components associated with the physical process. Several physics-based hydrological numerical simulation models, such as HEC-RAS [26], LISFLOOD [32], LISFLOOD-FP [6], are commonly used to simulate floods. However, such models are usually computation expensive and expert knowledge is required for both design and for accurate parameter tuning.

We consider the problem of hydrological modelling and simulation. Neural network models are known for their flexibility, efficient computation and capacity to deal with nonlinear correlation inside data. We propose to learn a flood surrogate model by training a neural network with pairs of inputs and outputs from the numerical model. We empirically demonstrate that the neural network can be used as a surrogate model to effectively and efficiently simulate the flood.

The neural network model that we train learns a general model. With the trained model from a given data set, the neural network is capable of simulating directly spatially different terrains. Moreover, while a neural network is generally constrained to a fixed size of its input, the model that we propose is able to simulate terrains of different sizes and spatial characteristics.

This paper is structured as follows. Section 2 summarises the main related works regarding physics-based hydrological and flood models as well as statistical machine learning models for flood simulation and prediction. Section 3 presents our methodology. Section 4 presents the data set, parameters setting and evaluation metrics. Section 5 describes and evaluates the performance of the proposed models. Section 6 presents the overall conclusions and outlines future directions for this work.

2 Related Work

Current flood models simulate the fluid movement by solving equations derived from physical laws with many hydrological process assumptions. These models can be classified into one-dimensional (1D), two-dimensional (2D) and three-dimensional (3D) models depending on the spatial representation of the flow. The 1D models treat the flow as one-dimension along the river and solve 1D Saint-Venant equations, such as HEC-RAS [1] and SWMM [25]. The 2D models receive the most attention and are perhaps the most widely used models for flood [28]. These models solve different approximations of 2D Saint-Venant equations. Two-dimensional models such as HEC-RAS 2D [9] is implemented for simulating the flood in Assiut plateau in southwestern Egypt [12] and Bolivian Amazonia [23]. Another 2D flow models called LISFLOOD-FP solve dynamic

wave model by neglecting the advection term and reduce the computation complexity [7]. The 3D models are more complex and mostly unnecessary as 2D models are adequate [28]. Therefore, we focus our work on 2D flow models.

Instead of a conceptual physics-based model, several statistical machine learning based models have been utilised [4,21]. One state-of-the-art machine learning model is the neural network model [27]. Tompson [29] uses a combination of the neural network models to accelerate the simulation of the fluid flow. Bar-Sinai [5] uses neural network models to study the numerical partial differential equations of fluid flow in two dimensions. Raissi [24] developed the physics informed neural networks for solving the general partial differential equation and tested on the scenario of incompressible fluid movement. Dwivedi [11] proposes a distributed version of physics informed neural networks and studies the case on Navier-Stokes equation for fluid movement.

Besides the idea of accelerating the computation of partial differential equation, some neural networks have been developed in an entirely data-driven manner. Ghalkhani [14] develops a neural network for flood forecasting and warning system in Madarsoo river basin at Iran. Khac-Tien [16] combines the neural network with a fuzzy inference system for daily water levels forecasting. Other authors [31,34] apply the neural network model to predict flood with collected gauge measurements. Those models, implementing neural network models for one dimension, did not take into account the spatial correlations. Authors of [18,35] use the combinations of convolution and recurrent neural networks as a surrogate model of Navier-Stokes equations based fluid models with a higher dimension.

The recent work [22] develops a convolutional neural network model to predict flood in two dimensions by taking the spatial correlations into account. The authors focus on one specific region in the Colorado River. It uses a convolutional neural network and a conditional generative adversarial network to predict water level at the next time step. The authors conclude neural networks can achieve high approximation accuracy with a few orders of magnitude faster speed.

Instead of focusing on one specific region and learning a model specific to the corresponding terrain, our work focuses on learning a general surrogate model applicable to terrains of different sizes and spatial characteristics with a data-driven machine learning approach.

3 Methodology

We propose to train a neural network with pairs of inputs and outputs from an existing flood simulator. The output provides the necessary supervision. We choose the open-source Python library Landlab, which is LISFLOOD-FP based. We first define our problem in Subsect. 3.1. Then, we introduce the general ideas of the numerical flood simulation model and Landlab in Subsect. 3.2. Finally, we present our solution in Subsect. 3.3.

3.1 Problem Definition

We first introduce the representation of three hydrological parameters that we use in the two-dimensional flood model. A digital elevation model (DEM) D is a $w \times l$ matrix representing the elevation of a terrain surface. A water level H is a $w \times l$ matrix representing the water elevation of the corresponding DEM. A rainfall intensity I generally varies spatially and should be a matrix representing the rainfall intensity. However, the current simulator assumes that the rainfall does not vary spatially. In our case, I is a constant scalar.

Our work intends to find a model that can represent the flood process. The flood happens because the rain drives the water level to change on the terrain region. The model receives three inputs: a DEM D, the water level H^t and the rainfall intensity I^t at the current time step t. The model outputs the water level H^{t+1} as the result of the rainfall I^t on DEM D. The learning process can be formulated as learning the function $\mathcal{L}$: $\mathbb{R}^{l\times w} \times \mathbb{R}^{l\times w} \times \mathbb{R} \rightarrow \mathbb{R}^{l\times w}$, which predicts $H^{t+1} = \mathcal{L}(D, H^t, I^t)$.

3.2 Numerical Flood Simulation Model and Landlab

Physics-driven hydrology models for the flood in two dimensions are usually based on the two-dimensional shallow water equation, which is a simplified version of Navier-Stokes equations with averaged depth direction [28]. By ignoring the diffusion of momentum due to viscosity, turbulence, wind effects and Coriolis terms [10], the two-dimensional shallow water equations include two parts: conservation of mass and conservation of momentum shown in Eqs. 1 and 2,

$$\frac{\partial h}{\partial t} + \frac{\partial hu}{\partial x} + \frac{\partial hv}{\partial y} = 0 \tag{1}$$

$$\begin{aligned} \frac{\partial hu}{\partial t} + \frac{\partial}{\partial x}\left(hu^2 + \frac{1}{2}gh^2\right) + \frac{\partial huv}{\partial y} + gh\frac{\partial Z(x,y)}{\partial x} + ghS_{f_x} = 0 \\ \frac{\partial hv}{\partial t} + \frac{\partial huv}{\partial x} + \frac{\partial}{\partial y}\left(hv^2 + \frac{1}{2}gh^2\right) + gh\frac{\partial Z(x,y)}{\partial y} + ghS_{f_y} = 0 \end{aligned} \tag{2}$$

where h is the water depth, g is the gravity acceleration, (u, v) are the velocity at x, y direction, $Z(x, y)$ is the topography elevation function and S_{f_x}, S_{f_y} are the friction slopes [33] which are estimated with friction coefficient η as

$$S_{f_x} = \frac{u\eta\sqrt{u^2+v^2}}{h^{4/3}},\ S_{f_y} = \frac{v\eta\sqrt{u^2+v^2}}{h^{4/3}}$$

For the two-dimensional shallow water equations, there are no analytical solutions. Therefore, many numerical approximations are used.

LISFLOOD-FP is a simplified approximation of the shallow water equations, which reduces the computational cost by ignoring the convective acceleration term (the second and third terms of two equations in Eq. 2) and utilising an explicit finite difference numerical scheme. The LISFLOOD-FP firstly calculate

the flow between pixels with mass [20]. For simplification, we use the 1D version of the equations in x-direction shown in Eq. 3,

$$q_{i+1/2,j}^{t+\Delta t} = q_{i+1/2,j}^{t} - gh_{i+1/2,j}^{t}\Delta t\Big(gh_{i+1/2,j}^{t}\frac{\partial Z(x,y)}{\partial x} + \frac{\eta^2 q_{i+1/2,j}^{t} q_{i+1/2,j}^{t+\Delta t}}{h_{i+1/2,j}^{t\ 10/3}}\Big) \quad (3)$$

where the $q_{i+1/2,j}^{t} = h_{i+1/2,j}^{t} \times u_{i+1/2,j}$ is the discharge per unit width in x-direction. The result of 1D can be directly transferable to 2D due to the uncoupled nature of those equations [3]. Then, for each pixel, its water level h is updated as Eq. 4,

$$\begin{aligned} h_{i,j}^{t+\Delta t} = h_{i,j}^{t} + \Delta t(&q_{i-1/2,j}^{t+\Delta t}/\Delta x \\ &+ q_{i,j-1/2}^{t+\Delta t}/\Delta y - q_{i+1/2,j}^{t+\Delta t}/\Delta x - q_{i,j+1/2}^{t+\Delta t}/\Delta y) \end{aligned} \quad (4)$$

To sum up, for each pixel at location i, j, the solution derived from LISFLOOD-FP can be written in a format shown in Eq. 5,

$$H_{i,j}^{t+1} = \Theta([H_{i+1,j}^{t}, H_{i-1,j}^{t}, H_{i,j+1}^{t}, H_{i,j-1}^{t}], I^t, D) \quad (5)$$

where $H_{i,j}^{t}$ is the water level at location i, j of time step t, or in general as $H^{t+1} = \Theta(D, H^t, I^t)$. However, the numerical solution as Θ is computationally expensive including assumptions for the hydrology process in flood. There is an enormous demand for parameter tuning of the numerical solution Θ once with high-resolution two-dimensional water level measurements mentioned in [36].

Therefore, we use such numerical model to generate pairs of inputs and outputs for the surrogate model. We choose the LISFLOOD-FP based open-source Python library, Landlab [2] since it is a popular simulator in regional two-dimensional flood studies. Landlab includes tools and process components that can be used to create hydrological models over a range of temporal and spatial scales. In Landlab, the rainfall and friction coefficients are considered to be spatially constant and evaporation and infiltration are both temporally and spatially constant. The inputs of the Landlab is a DEM and a time series of rainfall intensity. The output is a times series of water level.

3.3 Proposed Neural Network Model

We propose here that a neural network model can provide an alternative solution for such a complex hydrology dynamic process. Neural networks are well known as a collection of nonlinear connected units, which is flexible enough to model the complex nonlinear mechanism behind [19]. Moreover, a neural network can be easily implemented on general purpose Graphics Processing Units (GPUs) to boost its speed. In the numerical solution of the shallow water equation shown in Subsect. 3.2, the two-dimensional spatial correlation is important to predict the water level in flood. Therefore, inspired by the capacity to extract spatial correlation features of the neural network, we intend to investigate if a neural network model can learn the flood model $\mathcal{L}$ effectively and efficiently.

We propose a small and flexible neural network architecture. In the numerical solution Eq. 5, the water level for each pixel of the next time step is only correlated with surrounding pixels. Therefore, we use, as input, a 3×3 sliding window on the DEM with the corresponding water levels and rain at each time step t. The output is the corresponding 3×3 water level at the next time step $t+1$. The pixels at the boundary have different hydrological dynamic processes. Therefore, we pad both the water level and DEM with zero values. We expect that the neural network model learns the different hydrological dynamic processes at boundaries. One advantage of our proposed architecture is that the neural network is not restricted by the input size of the terrain for both training and testing. Therefore, it is a general model that can be used in any terrain size. Figure 1 illustrates the proposed architecture on a region with size 6×6.

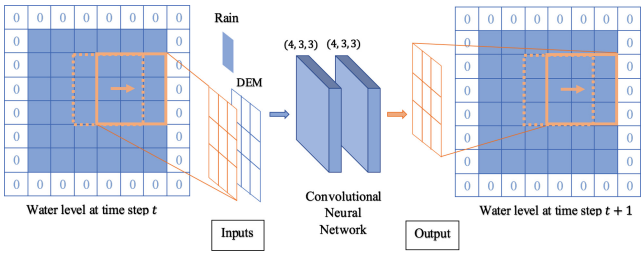


Fig. 1. Visualisation of the proposed architecture.

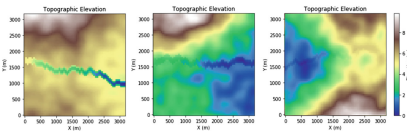
4 Performance Evaluation

In this Section, we empirically evaluate the performance of the proposed model. In Subsect. 4.1, we describe how to generate synthetic DEMs. Subsect. 4.2 presents the experimental setup to test our method on synthetic DEMs as a micro-evaluation. Subsect. 4.3 presents the experimental setup on the case in Onkaparinga catchment. Subsect. 4.4 presents details of our proposed neural network. Subsect. 4.5 shows the evaluation metrics of our proposed model.

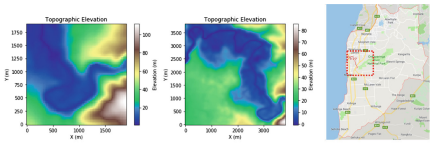
4.1 Generating Synthetic DEM

In order to generate synthetic DEMs, we modify Alexandre Delahaye's work[1]. We arbitrarily set the size of the DEMs to 64×64 and its resolution to 30 metres. We generate three types of DEMs in our data set that resembles real world terrains surface as shown in Fig. 2a, namely, a river in a plain, a river with a mountain on one side and a plain on the other and a river in a valley with mountains on both sides.

[1] https://beyondthemaps.wordpress.com/2017/11/01/random-dem-generator/, visited on 6th September 2019.



(a) Three types of synthetic DEM with size 64×64. Left: river on a plain, middle: river with a mountain on one side and a plain on another side, right: river in the valley with mountains on both sides.



(b) Two DEMs selected from Lower Onkaparinga river region. Left: size 64 × 64, middle: size 128 × 128, right: Lower Onkaparinga river region from Google Map.

Fig. 2. DEMs used for experiements

4.2 Experiments on Synthetic DEM

We evaluate the performance in two cases. In **Case 1**, the network is trained and tested with one DEM. This DEM has a river in the valley with mountains on both sides, as shown in Fig. 2a right. In **Case 2**, the network is trained and tested with 200 different synthetic DEMs.

The data set is generated with Landlab. For all the flood simulations in Landlab, the boundary condition is set to be closed on four sides. This means that rainfall is the only source of water in the whole region. The roughness coefficient is set to be 0.003. We control the initial process, rainfall intensity and duration time for each sample. The different initial process is to ensure different initial water level in the whole region. After the initial process, the system run for 40 h with no rain for stabilisation. We run the simulation for 12 h and record the water levels every 10 min. Therefore, for one sample, we record a total of 72 time steps of water levels. Table 1 summarises the parameters for generating samples in both **Case 1** and **Case 2**.

Table 1. Parameters for synthetic floods on **Case 1** and **Case 2**. The initial process contains two parts as (Initial rainfall intensity, duration time)

Parameter	Case 1	Case 2
	Value	Value
Initial process	(2,1), (2,2) (6,1), (6,2)	(6,1)
Flood rainfall intensity (mm/hr)	1 to 20	5,10,15
Flood duration time (hr)	1 to 12	3,6,9,12
DEM	1	200
Total number of samples	960	2400

4.3 Case of the Onkaparinga Catchment

The Onkaparinga catchment, located at Lower Onkaparinga river, south of Adelaide, South Australia, has experienced many notable floods, especially in 1935 and 1951. Many research and reports have been done in this region [15]. We get two DEM data with size 64×64 and 128×128 from the Australia Intergovernmental Committee on Surveying and Mapping's Elevation Information System[2]. Figure 2b shows the DEM of Lower Onkaparinga river. We implement the neural network model under three cases. In **Case 3**, we train and test on 64×64 Onkaparinga river DEM. In **Case 4**, we test 64×64 Onkaparinga river DEM directly with **Case 2** trained model. In **Case 5**, we test 128×128 Onkaparinga river DEM directly with **Case 2** trained model. We generate the data set for both 64×64 and 128×128 DEM from Landlab. The initial process, rainfall intensity and rain duration time of both DEM are controlled the same as in **Case 1**.

4.4 Neural Network Models

The architecture of the neural network model is visualized as in Fig. 1. It firstly upsamples the rain input into 3×3 and concatenates it with 3×3 water level input. Then, it is followed by several batch normalisation and convolutional layers. The activation functions are ReLU and all convolutional layers have the same size padding. The total parameters for the neural network are 169. The model is trained by Adam with the learning rate as 10^{-4}. The batch size for training is 8. The data set has been split with ratio 8:1:1 for training, validation and testing. The training epoch is 10 for **Case 1** and **Case 3** and 5 for **Case 2**.

We train the neural network model on a machine with a 3 GHz AMD Ryzen™ 7-1700 8-core processor. It has a 64 GB DDR4 memory and an NVIDIA GTX 1080Ti GPU card with 3584 CUDA cores and 11GB memory. The operating system is Ubuntu 18.04 OS.

4.5 Evaluation Metrics

In order to evaluate the performance of our neural network model, we use global measurements metrics for the overall flood in the whole region. These metrics are global mean squared error: $global\ MSE_t = \frac{1}{mn}\sum_m \sum_n [P_t(i,j) - T_t(i,j)]^2$, global mean absolute percentage error: $global\ MAPE_t = \frac{1}{mn}\sum_m \sum_n |\frac{P_t(i,j) - T_t(i,j)}{T_t(i,j)}|$, where $P_t(i,j), T_t(i,j)$ is the predicted and the ground truth water level at location i, j at the t-th time step. We measure the global MSE and the global MAPE both at each time step and their mean for all time steps.

From the hydrological point of view, some areas are more important for observation and model calibration. For instance, the exit point of river or a watershed exit. Therefore, we propose two local measurements metrics at the watershed exit: local mean squared error: $local\ MSE = \frac{1}{steps}\sum_t [P_t - T_t]^2$, local mean absolute percentage error: $local\ MAPE = \frac{1}{steps}\sum_t |\frac{P_t - T_t}{T_t}|$, where P_t, T_t

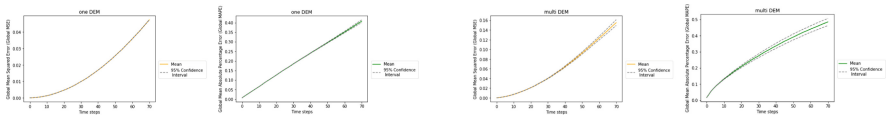
[2] https://elevation.fsdf.org.au/, visited on 30th September 2019.

are the means of predicted and ground truth water level of pixels near watershed exit at t-th time step. For all evaluation metrics the lower the value, the better.

5 Results and Discussion

5.1 Results

Experiments on Synthetic DEM. In this section, we show the results of our neural network model on **Case 1** and **Case 2**. For global performance, our model shows increasing trend of MSE and MAPE in both **Case 1** and **Case 2** according to Fig. 3c and 3f. Our model yields higher mean and variance of MAPE under **Case 2** according to Table 2a. For both cases, the maximum MAPE at first 6 h is less than 25%, the maximum MAPE at 12 h is less than 50%. For local performance, **Case 2** has lower mean of MAPE in Table 2a. Thus, our model works effectively on both one DEM and multiple DEMs cases.



(a) Global MSE (b) Global MAPE (d) Global MSE (e) Global MAPE

(c) The global performance metrics for each time step of our neural network models under **Case 1**.

(f) The global performance metrics for each time step of our neural network models under **Case 2**.

Fig. 3. The global performance metrics for each time step of our neural network models under **Case 1** and **Case 2**.

Table 2. The mean of evaluation metrics over all time steps **Case 1** to **Case 5**.

(a) The mean of evaluation metrics over all time steps under **Case 1** and **Case 2**.

			Case 1	**Case 2**
Global	**MSE**	Mean	0.0168	0.0606
		Var	0.0024	0.0232
	MAPE	Mean	0.2108	0.2920
		Var	0.0254	0.0742
Local	**MSE**	Mean	0.1726	1.0667
		Var	0.0001	0.7122
	MAPE	Mean	0.2468	0.1652
		Var	0.00001	0.0172

(b) The mean of evaluation metrics over all time steps under **Case 3**, **Case 4** and **Case 5**.

			Case 3	**Case 4**	**Case 5**
Global	**MSE**	Mean	0.0577	0.0529	0.0516
		Var	0.0072	0.0250	0.0212
	MAPE	Mean	1.0542	0.5321	0.5254
		Var	0.1267	0.1404	0.1317
Local	**MSE**	Mean	0.1171	0.2137	0.2014
		Var	0.00001	0.0247	0.0121
	MAPE	Mean	0.1630	0.2149	0.3445
		Var	0.00001	0.0121	0.0162

Table 3. Time consumption of the neural network model and Landlab for floods.

	Case 1	Case 2	Case 3	Case 4	Case 5	Landlab	
						Size 64×64	Size 128×128
Train	4.05 h	7.00 h	4.22 h			47 s	8 min
Test	1.20 s	2.10 s	1.13 s	2.04 s	4.23 s		

Case of the Lower Onkaparinga River. In this section, we show the results of our neural network model on the case of the Lower Onkaparinga river (**Case 3** and **Case 4**). For global performance, the mean of MSE and MAPE in **Case 4** is lower than in **Case 3**, while the variance is higher in **Case 4** than in **Case 3** according to Table 2b. For local performance, **Case 3** is better than **Case 4** in all metrics according to Table 2b. Thus, without retraining the existing model, the trained neural network from **Case 2** can be applied directly on new DEM with a good global performance.

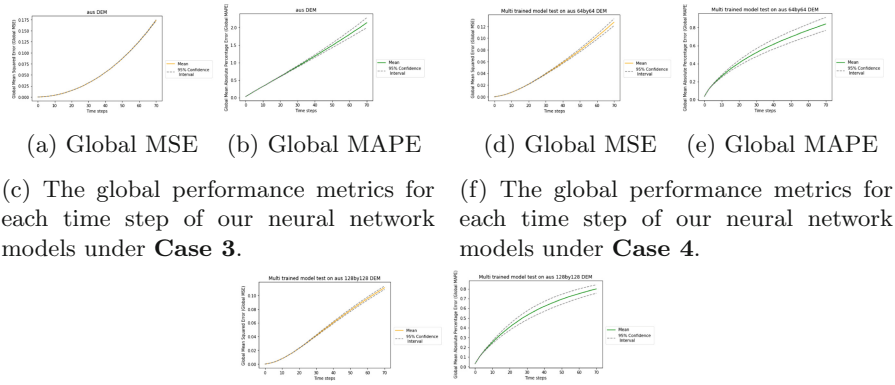
Case 5 is to test the scalability of our model for the different size DEM. In Table 2b, for global performance, the MAPE of **Case 5** is around 50% less than both **Case 3** and **Case 4**, and for local performance, the MAPE of **Case 5** is 34.45%. Similarly, without retraining the existed model, the trained neural network from **Case 2** can be applied directly on DEM with different size with a good global performance.

5.2 Efficiency

We present the time needed for the flood simulation of one sample in Landlab and in our neural network model (without the training time) in Table 3. The average time of the neural network model for a 64×64 DEM is around 1.6 s, while it takes 47 s in Landlab. Furthermore, for a 128×128 DEM, Landlab takes 110 more time than the neural network model. Though the training of the neural network model is time consuming, it can be reused without further training or tuning terrains of different sizes and spatial characteristics. It remains effective and efficient (Fig. 4).

6 Conclusion and Future Work

We propose a neural network model, which is trained with pairs of inputs and outputs of an off-the-shelf numerical flood simulator, as an efficient and effective general surrogate model to the simulator. The trained network yields a mean absolute percentage error of around 20%. However, the trained network is at least 30 times faster than the numerical simulator that is used to train it. Moreover, it is able to simulate floods on terrains of different sizes and spatial characteristics not directly represented in the training. We are currently extending our work to take into account other meaningful environmental elements such as the land coverage, geology and weather.



(a) Global MSE (b) Global MAPE (c) The global performance metrics for each time step of our neural network models under **Case 3**.

(d) Global MSE (e) Global MAPE (f) The global performance metrics for each time step of our neural network models under **Case 4**.

(g) Global MSE (h) Global MAPE (i) The global performance metrics for each time step of our neural network models under **Case 5**.

Fig. 4. The global performance metrics for each time step of our neural network models under **Case 3**, **Case 4** and **Case 5**.

Acknowledgment. This work is supported by the National University of Singapore Institute for Data Science project WATCHA: WATer CHallenges Analytics. Abhishek Saha is supported by National Research Foundation grant number NRF2017VSG-AT3DCM001-021.

References

1. HEC-RAS river analysis system, user's manual, version 2.2 (1998)
2. Adams, J.M., et al.: The Landlab v1. 0 OverlandFlow component: a Python tool for computing shallow-water flow across watersheds. Geosci. Model Dev. **10**(4), 1645–1663 (2017)
3. de Almeida, G.A., Bates, P., Freer, J.E., Souvignet, M.: Improving the stability of a simple formulation of the shallow water equations for 2-D flood modeling. Water Resour. Res. **48**(5) (2012)
4. Asher, M.J., Croke, B.F., Jakeman, A.J., Peeters, L.J.: A review of surrogate models and their application to groundwater modeling. Water Resour. Res. **51**(8), 5957–5973 (2015)
5. Bar-Sinai, Y., Hoyer, S., Hickey, J., Brenner, M.P.: Learning data-driven discretizations for partial differential equations. Proc. Natl. Acad. Sci. **116**(31), 15344–15349 (2019)
6. Bates, P.D., De Roo, A.: A simple raster-based model for flood inundation simulation. J. Hydrol. **236**(1–2), 54–77 (2000)
7. Bates, P.D., Horritt, M.S., Fewtrell, T.J.: A simple inertial formulation of the shallow water equations for efficient two-dimensional flood inundation modelling. J. Hydrol. **387**(1–2), 33–45 (2010)
8. Beven, K.J.: Rainfall-Runoff Modelling: the Primer. Wiley, Hoboken (2011)

9. Brunner, G.: HEC-RAS river analysis system hydraulic userś manual (2008)
10. Delis, A.I., Katsaounis, T.: Numerical solution of the two-dimensional shallow water equations by the application of relaxation methods. Appl. Math. Model. **29**(8), 754–783 (2005)
11. Dwivedi, V., Parashar, N., Srinivasan, B.: Distributed physics informed neural network for data-efficient solution to partial differential equations. arXiv preprint arXiv:1907.08967 (2019)
12. Ezz, H.: Integrating gis and HEC-RAS to model assiut plateau runoff. Egypt. J. Remote Sens. Space Sci. **21**(3), 219–227 (2018)
13. Gaume, E., Payrastre, O.: Flood hydrology processes and their variabilities. In: Floods, pp. 115–127. Elsevier (2017)
14. Ghalkhani, H., Golian, S., Saghafian, B., Farokhnia, A., Shamseldin, A.: Application of surrogate artificial intelligent models for real-time flood routing. Water Environ. Journal. **27**(4), 535–548 (2013)
15. Hill, P., Daniell, T., et al.: Extreme flood estimation-guesses at big floods? Water Down Under 94: Surface Hydrology and Water Resources Papers, p. 193 (1994)
16. Khac-Tien Nguyen, P., Hock-Chye Chua, L.: The data-driven approach as an operational real-time flood forecasting model. Hydrol. Process. **26**(19), 2878–2893 (2012)
17. Khan, A.N., et al.: Analysis of flood causes and associated socio-economic damages in the Hindukush region. Nat. Hazards **59**(3), 1239 (2011)
18. Kim, B., Azevedo, V.C., Thuerey, N., Kim, T., Gross, M., Solenthaler, B.: Deep fluids: a generative network for parameterized fluid simulations. Comput. Graph. Forum **38**(2), 59–70 (2019)
19. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3431–3440 (2015)
20. Neal, J., Dunne, T., Sampson, C., Smith, A., Bates, P.: Optimisation of the two-dimensional hydraulic model LISFOOD-FP for CPU architecture. Environ. Model. Softw. **107**, 148–157 (2018)
21. Oyebode, O., Stretch, D.: Neural network modeling of hydrological systems: a review of implementation techniques. Nat. Resour. Model. **32**(1), e12189 (2019)
22. Qian, K., Mohamed, A., Claudel, C.: Physics informed data driven model for flood prediction: application of deep learning in prediction of urban flood development. arXiv preprint arXiv:1908.10312 (2019)
23. Quirogaa, V.M., Kurea, S., Udoa, K., Manoa, A.: Application of 2D numerical simulation for the analysis of the February 2014 Bolivian Amazonia flood: application of the new HEC-RAS version 5. Ribagua **3**(1), 25–33 (2016)
24. Raissi, M., Perdikaris, P., Karniadakis, G.E.: Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. J. Comput. Phys. **378**, 686–707 (2019)
25. Rossman, L.: Storm water management model-user's manual v. 5.0. US Environmental Protection Agency: Cincinnati, Ohio, USA (2009)
26. Scharffenberg, W., Harris, J.: Hydrologic engineering center hydrologic modeling system, HEC-HMS: interior flood modeling. In: World Environmental and Water Resources Congress 2008: Ahupua'A, pp. 1–3 (2008)
27. Sit, M., Demir, I.: Decentralized flood forecasting using deep neural networks. arXiv preprint arXiv:1902.02308 (2019)
28. Teng, J., Jakeman, A.J., Vaze, J., Croke, B.F., Dutta, D., Kim, S.: Flood inundation modelling: a review of methods, recent advances and uncertainty analysis. Environ. Model. Softw. **90**, 201–216 (2017)

29. Tompson, J., Schlachter, K., Sprechmann, P., Perlin, K.: Accelerating Eulerian fluid simulation with convolutional networks. In: Proceedings of the 34th International Conference on Machine Learning, vol. 70, pp. 3424–3433. JMLR. org (2017)
30. UNISDR: UNISDR annual report 2018 (2018)
31. Valipour, M., Banihabib, M.E., Behbahani, S.M.R.: Comparison of the ARMA, ARIMA, and the autoregressive artificial neural network models in forecasting the monthly inflow of Dez dam reservoir. J. Hydrol. **476**, 433–441 (2013)
32. Van Der Knijff, J., Younis, J., De Roo, A.: Lisflood: a GIS-based distributed model for river basin scale water balance and flood simulation. Int. J. Geogr. Inf. Sci. **24**(2), 189–212 (2010)
33. Vreugdenhil, C.B.: Numerical Methods for Shallow-Water Flow, vol. 13. Springer, Heidelberg (2013). https://doi.org/10.1007/978-94-015-8354-1
34. Wang, J.H., Lin, G.F., Chang, M.J., Huang, I.H., Chen, Y.R.: Real-time water-level forecasting using dilated causal convolutional neural networks. Water Resour. Management. **33**, 1–22 (2019)
35. Wiewel, S., Becher, M., Thuerey, N.: Latent space physics: towards learning the temporal evolution of fluid flow. Comput. Graph. Forum **38**(2), 71–82 (2019)
36. Zhang, Z., Zhou, Y., Liu, H., Gao, H.: In-situ water level measurement using NIR-imaging video camera. Flow Meas. Instrum. **67**, 95–106 (2019)