www.ijecs.in
International Journal Of Engineering And Computer Science
Volume 10 Issue 11 November 2021, Page No. 25420-25430
ISSN: 2319-7242 DOI: 10.18535/ijecs/v10i11.4636

A coupled models Hydrodynamics - Multi headed Deep convolutional neural network for rapid forecasting large-scale flood inundation

Dr. Sofiane HADJI

Scientific Director, SIXENSE Engineering (Vinci Group), 22 rue Lavoisier - 92000 Nanterre, France
Email: sofiane.hadji@sixense-group.com

Abstract: Modeling large-scale flood inundation requires weeks of calculations using complex fluid software. The state-of-the-art in operational hydraulic modeling does not currently allow flood real-time forecasting fields. Data driven models have small computational costs and fast computation times and may be useful to overcome this problem. In this paper, we propose a new modeling approach based on a coupled of Hydrodynamics finite element model and Multi-headed Deep convolutional neural network (MH-CNN) with rain precipitations as input to forecast rapidly the water depth reached in large floodplain with few hours-ahead. For this purpose, one first builds a database containing different simulations of the physical model according to several rain precipitation scenarios (historic and synthetic). The multi-headed convolutional neural network is then trained using the constructed database to predict water depths. The pre-trained model is applied successfully to simulate the real July 2014 flood inundation in an 870 km^2 area of *La Nive* watershed in the south west of France. Because rain precipitation forecast data is more accessible than discharge one, this approach offers great potential for real-time flood modelling for ungauged large-scale territories, which represent a large part of floodplain in the world.

Keywords: Rapid flood forecasting; Large-scale flood inundation, Deep learning, Convolutional neural network; Rainfall, hydrodynamics model.

1. Introduction

Floods occur on a more frequent base than ever before. Due to climate change weather patterns change and rain intensity increases. Climate change also influences rainfall events. This causes increasing rainfall to flood more often. The increasing amount of rain is problematic especially in urban areas, which drainage system can often not handle this large amount in a short time.

Two-dimensional hydrodynamic models are widely used tools for simulating flows rivers and floodplain inundation phenomena. In these models, the flood propagation is generally described by the two-dimensional shallow water equations (2D), which must be solved using an appropriate numerical technic. Different methods exist for solving the large-scale environmental flow simulations problem, one can cite for example the Saint-Venant hyperbolic partial differential equations solved by explicit finite elements method (FEM) or finite difference method on single-node multi-GPU architectures [1, 2]. Another technic to solve this problem is the lattice Boltzmann method (LBM). It is a relatively recent technique which is able to approximate the Navier-Stokes equations by a collision-propagation scheme [3]. Lattice Boltzmann method however differs from standard approaches as finite element method (FEM) by its mesoscopic approach. It is an interesting alternative which is able to simulate complex phenomena on complex geometries. Its high parallelization makes also this method attractive in order to perform simulations on parallel hardware. Moreover, the emergence of high-performance computing (HPC) architectures using GPUs is also a great interest for many researchers [4]. Although physical models showed great capabilities for predicting a diverse range of flooding scenarios, they often require intensive computation, which prohibits short-term prediction. This makes the use of the hydraulic model for real-time simulations nearly infeasible, specifically for

emergency responses.

The advanced data-driven models such as Machine learning or Deep learning also have a tradition in flood modeling, which recently gained more popularity. Data-driven methods of prediction assimilate the measured climate indices and hydrometeorological parameters to provide better insight. A further reason for the popularity of such models is that they can numerically formulate the flood nonlinearity, solely based on historical data without requiring knowledge about the underlying physical processes. Another reason for using those methods is the easiest implementation with low computation cost, as well as fast training, validation, testing, and evaluation, with high performance compared to physical models, and relatively less complexity [5, 6, 7]. Nonetheless, those advanced data-driven algorithms have important drawback: If the data history is not sufficient or does not cover varieties of the task, their learning falls short, and hence, they cannot perform well when they are put into work. Therefore, using robust data enrichment is essential through for example numerical simulations with physical models.

The purpose of present study is to investigate the efficiency of a coupled Hydrodynamic - MH-CNN model to solve and predict rapidly the maximum water depths reached for large flood event in *La Nive* watershed in France. The modeled domain covers an area of approximately 870 km^2 and meshed by more than million finite elements nodes.

The coupled hydrodynamics-Deep learning focused method suggested in this study can help specialist engineers make more accurate and effective predictions. This proposed study may be considered as a part of creative prediction approaches and stochastic views in hydrology.

2. Hydrodynamic flow model

The two-dimensional finite element hydrodynamical model developed in this study was based on the resolution of shallow-water equations, which were obtained by using hydrostatic and *Boussinesq* approximations and by integrating *Navier-Stokes* equations over total water depth. Notice, that for many practical surface-water flow applications, knowledge of the full three-dimensional flow behaviors is not needed, and it is enough to use the depth average flow quantities in two perpendicular horizontal directions. Thus, the depth-averaged momentum, and continuity equations lead to the following *Saint-Venant* set of equations:

$$\begin{cases} \frac{\partial \boldsymbol{u_f}}{\partial t} + \boldsymbol{u_f}.\nabla(\boldsymbol{u_f}) + g\,\nabla h - \mu\,\nabla \boldsymbol{u_f} = \mathbf{F} \\ \frac{\partial h}{\partial t} + \nabla(h\boldsymbol{u_f}) = r \end{cases} \quad (1)$$

where $\mathbf{u}_f = (u_f, v_f)$ the depth average cartesian velocity with u_f and v_f are the components in the x and y coordinate directions, defined by:

$$\boldsymbol{u_f} = \frac{1}{H}\int_{z_b}^{h}(u,v)\,dz \quad (2)$$

H (x, y, t) is the water depth, $z_b(x, y, t)$ the bed elevation of the river and $h(x, y, t) = z_b(x, y, t) + H$ (x, y, t) is the water surface elevation. μ is the effective viscosity, which includes the dispersion and the turbulence contributions. $\mathbf{F} = (Fx, Fy)$ integrate volume forces (*Coriolis*), actions exerted on the bottom (friction) or on the free surface (wind) and forces as radiation constraints. r is a source term due to rainfall. Thus:

$$\mathbf{F} = -\beta\,\boldsymbol{u_f} + 2\,\boldsymbol{u_f}\boldsymbol{k}\,\omega\sin\varphi - s\frac{\rho_a}{\rho}\frac{1}{H}\|\boldsymbol{W}\|\boldsymbol{W} + \frac{1}{H}\nabla(S_x) \quad (3)$$

where $\boldsymbol{k} = (0, 0, 1)$ is the vertical unit vector; ω is the earth velocity in *rd/sec*, φ the latitude in degrees, $\boldsymbol{s}$ is a coefficient equal to 0.0026, $\boldsymbol{W} = (w_x, w_y)$ are wind components in x and y directions, $S_x = (s_{xx}, s_{xy})$ are radiation constraints components in x and y directions early calculated by a water waves model. ρ_a and ρ are densities of air and water respectively. β depends on the value of the following coefficient γ defined by:

$$\gamma = \left[19.8\,Ln\left(\frac{915}{\mu}\right)\right]^{-1} \text{thus}: \quad \begin{cases} \beta = \frac{g}{\gamma^2}\frac{\sqrt{uf^2}}{H} & if\ c > 1 \\ \beta = g\gamma^2\frac{\sqrt{uf^2}}{H^{3/4}} & if\ c < 1 \end{cases}$$

To satisfy the *Ladyzhenskaya-Babuška-Brezzi* (LBB) condition and avoid numerical instability and spurious oscillations, we develop a P2-P1 element for finite element approximation: A triangle is composed of 4 sub-triangles; velocity field is linear on each sub-triangle and free surface field is linear on the base triangle (Figure 1).

Over each sub-triangle:

$$(u,v) = \sum N_i.(u,v)_i \quad (u^2, uv, v^2) = \sum N_i.(u^2, uv, v^2)_i$$

Over each base triangle:

$$h = \sum N_i\ h_i$$

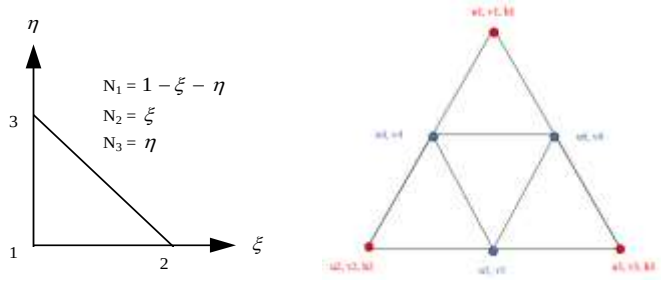


Figure 1: P2-P1 element: left, reference triangle for sub-triangle; right, 6 nodes base triangle composed of 4 sub-triangles

3. Multi-Headed CNN model

During the last decade, Convolutional Neural Networks (CNNs) have become the *de facto* standard for various Machine Learning operations. CNNs are feed-forward Artificial Neural Networks (ANNs) with alternating convolutional and subsampling layers. Deep 2D-CNNs with many hidden layers and millions of parameters have the ability to learn complex objects and patterns providing that they can be trained on a massive size visual database with ground-truth labels. With a proper training, this unique ability makes them the primary tool for various engineering applications for 2D signals such as images and video frames. Yet, this may not be a viable option in numerous applications over 1D signals especially when the training data is scarce or application specific.

To address this issue, 1D-CNNs have recently been proposed and immediately achieved the state-of-the-art performance levels in several applications such as personalized biomedical data classification and early diagnosis, structural health monitoring (SHM), anomaly detection and identification in power electronics and electrical motor fault detection [8, 9, 10]. Another major advantage is that a real-time and low-cost hardware implementation is feasible due to the simple and compact configuration of 1D-CNNs that perform only 1D convolutions (scalar multiplications and additions). CNNs share the same characteristics and follow the same approach, no matter if it is 1D, 2D or 3D. The key difference is the dimensionality of the input data and how the feature detector (or filter) slides across the data.

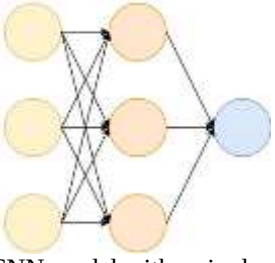


Figure 2: CNN model with a single output head

Multi-head neural networks or multi-head deep learning models are also known as multi-output deep learning models. The Multi-headed 1D-CNN presented in figure 3, is a more elaborate architecture to model our problem. This architecture consists of applying the 1D-CNN to each of the input sequences separately, the outputs of each of the sub models will be combined before making the prediction. This model offers more flexibility and better performance compared to a single head 1D-CNN presented in figure 2. For example, it allows to configure each sub-model differently for each input sequence such as the number of filters or the kernel size.

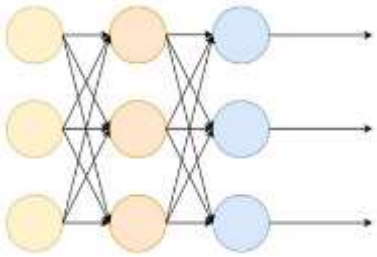


Figure 3: CNN model having multiple output heads

In this work we propose a multi-head Convolutional Neural Network (MH-CNN) architecture trained to predict rapidly with more accuracy the water depth in *La Nive* watershed. The proposed CNN model is developed in Python programming language using *Keras* module within the *Tensorflow 2.1* framework. The sequential application programming interface *Spyder* 5.0.5 (The Scientific Python Development Environment) is used to build the model, layer-by layer. The core structure of the MH-CNN model developed is graphically illustrated in figure 4.

Each of the sub model network has five hidden layers, including two convolutional layers and three dense layers. The dense layers are fully connected and act like a multi-layer perceptron (MLP) network. The output layer contains nodes equal to the number of finite elements nodes in the simulation domain (i.e. 1,072,000 for the current case study), and the input layer receives the different rainfall curve values from different scenarios (historical and synthetical curves). In addition to the MH-CNN, a simple CNN flood modelling method is also considered and constructed herein for comparison.

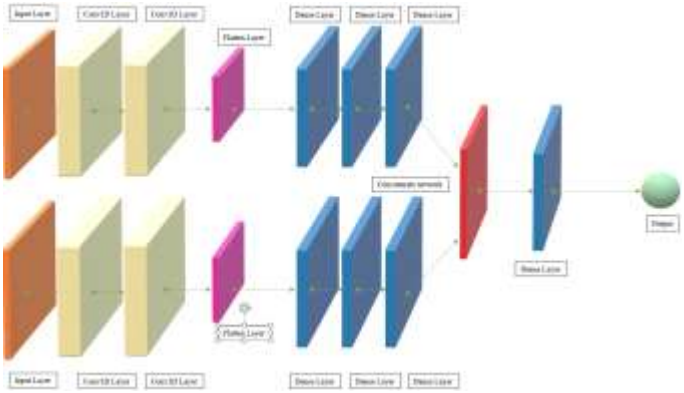


Figure 4: Multi-Headed CNN architecture for water depth time series forecasting

The hyperparameters of the proposed model were found using a Bayesian optimization approach. The procedure consists in launching a succession of calculation by changing and test for each time the parameters values. The best set of parameters are those gives the optimal performance measured on the validation data. The approach is the same as that proposed in [5].

Table 1: Hyperparameters of the proposed model

Model	Filter (size)	Neurons	Kernel (size)	Optim.	Batch (size)
CNN1	32-64	128-64-32	6		
CNN2	32-64		6		
Merge: CNN1 + CNN2		512		*Adam*	20

The *ReLU* (*Rectified Linear Unit*) is used as the activation function to increase the feature expression ability of the model. One selects the « mse » as loss function to optimize the multiple fully connected heads.

4. Study area

La Nive watershed is located in the west of the Atlantic Pyrenees. It covers more than 100,000 ha. It is composed by 53 municipalities with 97,600 inhabitants. The area is about 110,100 ha (i.e. more than 14% of the departmental area). The altitude varies from 5m to 1472 m and it contain 1,300 km of watercourses, including 365 km of watercourses over 10 km. 66,460 ha of the area are forests and semi-natural environments (67% of the territory).

Figure 5: Study domain

Like most other Pyrenean rivers, *La Nive* is a very abundant river. Its flow was observed over a period of 42 years (1967-2008), in *Cambo-les-Bains*, a small city located about fifteen kilometers from its confluence with the *Adour* river. The surface area thus observed is 870 km^2 or 85% of the entire watershed of the river. The interannual hydrological flow of the river at *Cambo-les-Bains* station is 30.2 m^3/s.

Figure 6: Average flow in m^3/s at *Cambo-les-Bains* station

At low water, the minimum consecutive volume for 3 days can drop to 4.8 m3/s, in the event of a dry five-year period, i.e. 4,800 liters per second, which is far from being severe, and rather normal compared to the average of the Pyrenean rivers in the basin. Floods can be very significant, especially as the size of the watershed is relatively large. The QIX 2 (maximum instantaneous flow of biennial flood) and quinquennial are respectively 390 and 530 m³/s. The QIX 10 (decennial flood) is 620 m³/s, the QIX 20 (20-year event)is 710 m³/s, while the QIX 50 (50-year event) is 820 m³/s. This is a probability of occurrence, not an average of occurrence. *La Nive* basin depends strongly on the rain precipitation.

5. Methodology

The methodology to solve the predicting model is presented as follow:

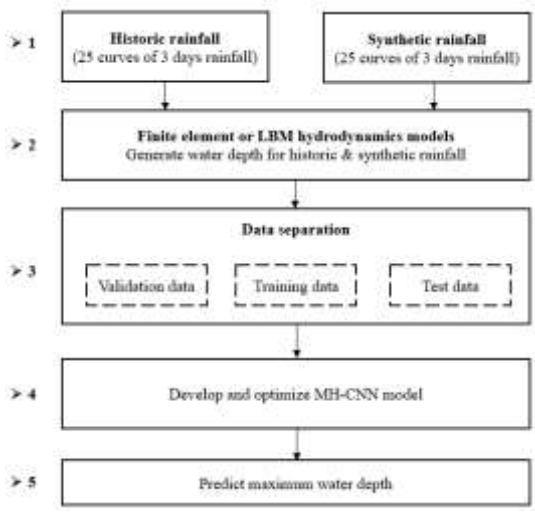
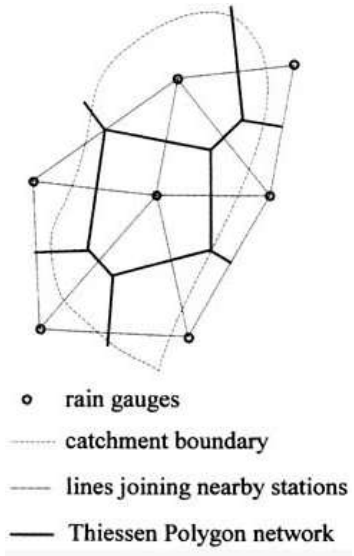


Figure 7: General structure of the proposed model

1. The rainfall stations on the ground provide punctual observations of the rains. One needs to know the rain precipitation on the basin: It is therefore necessary to estimate the rain at any point from a (small) number of point observations. Different interpolation methods are used. In our study, one uses the Thiessen's polygon method. The Thiessen polygon in figure 8 is a commonly used methodology for computing the mean areal precipitation for a catchment from rain gauge observations. The Thiessen method is based on the assumption that measured amounts at any station can be applied halfway to the next station in any direction, which means that for any point rainfall is equal to the observed rainfall at the closest gauge. The weights of the rain gauges are computed by their relative areas (A_i), which are estimated with the Thiessen polygon network.



$$R_{Mean} = \sum_{i=1}^{N} \alpha_i \, R_i$$

$$\alpha_i = \frac{A_i}{\sum_{j=1}^{N} A_j}$$

Figure. 8: Thiessen polygon for the Mean rainfall estimation

The historical curves are obtained by extracting from the curve of 2014 rainfall evolution (figure 9), 25 precipitation curves with a duration of 3 days each. The extraction is done in a random manner and include the July event from 3 to 5 July.

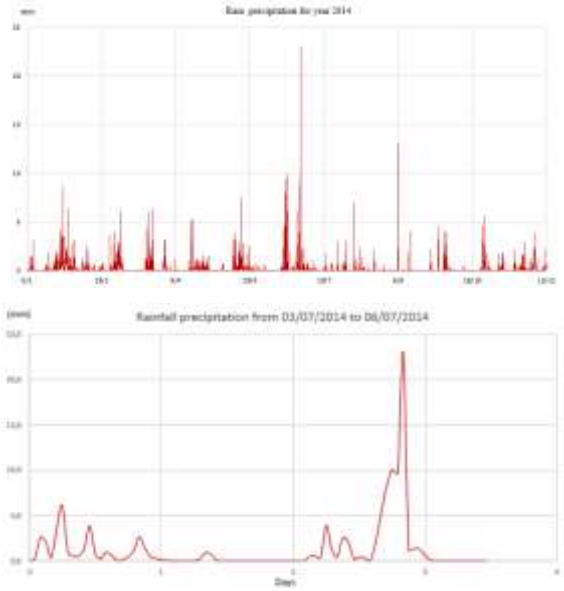


Figure 9: Historic rain precipitation for year 2014 and for July 2014 event witch cause strong damage

The synthetic curves are generated to create precipitation scenarios that have never happened but that allow better extrapolation by neural networks. For this purpose, one extract 25 other precipitation curves with 3 days duration each from the 2014 historic rainfall and adjust various peaks using the following formula [5] to increase their magnitudes as appropriate:

$$\boldsymbol{r}_i = r_{his} \cdot \frac{r_{peak}}{r_{max}} \quad where\ r_{peak} > r_{max} \qquad (5)$$

in which $\boldsymbol{r}_i$ is the synthetic rain precipitation, $\boldsymbol{r}_{peak}$ is the user specified peak, $r_{his}$is the historic rainfall and $r_{max}$is the historic peak rainfall. This generates the synthetic rainfall as shown in figure 9.

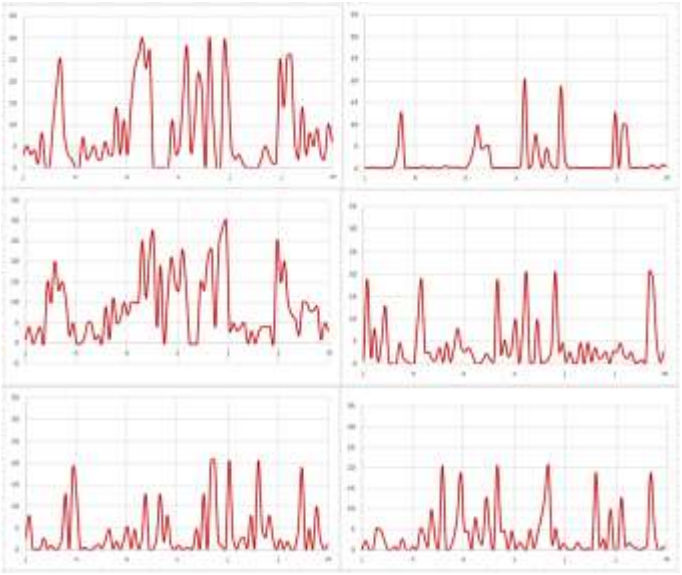


Figure 9: Examples of synthetic rain precipitation with 3 days duration each

2. The finite element hydrodynamics model based on equations 1, 2, 3 and 4 is then used to generate training and validation samples for the data-driven predictive models considered in this work. All the 50 (25 historic and 25 synthetic) rain precipitation curves were introduced as input rainfall sources to generate the different scenarios stocked in the database.

3. To evaluate the results of the prediction model, the database is divided into two sets: the training set and the test set. The test set is exclusively used to evaluate the model. For the evaluation of the network, the *root mean squared error* (RMSE), the *Nash-Sutcliffe Efficiency* coefficient (NSE) and the coefficient of determination R^2 of the predicted and measured water levels are used. The RMSE is defined through equation (6):

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (y_{obs} - y_{pred})^2}{n}} \quad (6)$$

where n is the number of nodes in the finite element mesh, y_{obs} and y_{pred} are the *'observed'* and *'predicted'* depth water values, respectively. RMSE represents the standard deviation of the differences between the values predicted by numerical model and those by MH-CNN model. RMSE = 0 returns a perfect fit between the predicted and the observed data.

The NSE coefficient is defined through Equation (7):

$$NSE = 1 - \frac{\sum_{i=1}^{n} (y_{obs} - y_{pred})^2}{\sum_{i=1}^{n} (y_{obs} - y_{mean})^2} \quad (7)$$

The mean of all water levels from the pre-calculated scenarios is written as y_{mean}. NSE = 1 represents a perfect fit between the reference and predicted data.

The coefficient of determination R^2 is a measure of the goodness of fit of a statistical model.

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (y_{obs} - \bar{y}_{pred})^2}{\sum_{i=1}^{n} (y_{obs} - y_{mean})^2} \quad (8)$$

Where $\bar{y}_{pred}$ are the predicted values from a statistical model and y_{mean} is the mean of observed values of the variables.

4. As presented in figure 4, the Multi-Headed-CNN is presented in the pseudo code in Algorithm 1.

Algorithm 1 : Construct and merge two sub model

1: Split from database and extract training data
x1_train, y1_train
x2_train, y2_train

2: Split from database and extract validation data
x1_val, y1_ val
x2_ val, y2_ val

3: Construct the multi-headed model
3.1: First neural network
visible1 = Input (shape = (steps, features))
cnn1 = Conv1D ( )(visible1)
cnn1 = Conv1D ( )(cnn1)
cnn1 = Flatten () (cnn1)
cnn1 = Dense ( )(cnn1)
cnn1 = Dense ( )(cnn1)
cnn1 = Dense ( )(cnn1)

3.2: Second neural network
visible2 = Input (shape = (steps, features))
cnn2 = Conv1D ( )(visible2)
cnn2 = Conv1D ( )(cnn2)
cnn2 = Flatten () (cnn2)
cnn2 = Dense ( )(cnn2)
cnn2 = Dense ( )(cnn2)
cnn2 = Dense ( )(cnn2)

4: Merge the two input sub-models
merge = concatenate ([cnn1, cnn2])
dense = Dense ( )(merge)
output = Dense (outputs) (dense)
model=Model(inputs=[visible1,visible2],outputs= output)
model.fit([x1_train,x2_train],[y1_train,y2_train],
validation_data=([x1_val,x2_val],[y1_val,y2_val]))

The hyperparameters of the proposed model were found using a Bayesian optimization approach. The pseudo code in Algorithm 2 show how it was done:

Algorithm 2 : Minimize model for given data and implicit hyperparameters

1: Construct the model with different parameters
Model=Sequential ()
model.add(Conv1D(choice([10,16,32,64,128,256,512]),
kernel_size=choice([1,2,3,4,5,6,7,8,9,10]))

```
model.add(Conv1D(choice([10,16,32,64,128,256,512]),
kernel_size=choice([1,2,3,4,5,6,7,8,9,10]))
model.add(Flatten())
model.add(Dense(choice([10,16,32,64,128,256,512]))
model.add(Dense (choice([10,16,32,64,128,256,512]))
model.add(Dense(choice [10,16,32,64,128,256,512]))
model.add(Dense(outputs))
model.compile(loss='mse',optimizer='adam')
result=model.fit(x_train,y_train,batch_size=choice([10,
20,30,40,50,80]))
```

2: Optimize and output the optimized hyperparameters

```
optim.minimize(model)
```

6. Application

The developed MH-CNN model for maximum water level prediction was tested in *La Nive* watershed which represent a surface of 870 km^2. From the results of our finite element hydrodynamic model, only the water level was considered, although more information would in principle be available. Hydrometric and topographic data are required to set up and run the hydrodynamic model for predicting large flood inundation. The initial digital elevation model (DEM) was a 5 m grid downloaded from SRTM DEM [12]. One could use this DEM to obtain a very precise simulation results, but that requires a mesh of more than 30 million finite element nodes, which is inconceivable. For our study, the spatial resolution taken was of 30 m. It is effectively wide to capture all the details of the flood, but this was considered sufficient to reproduce the dynamics of floods. Remember that the main goal is to obtain the peak of the water reached during a flood as well as their spatial positions.

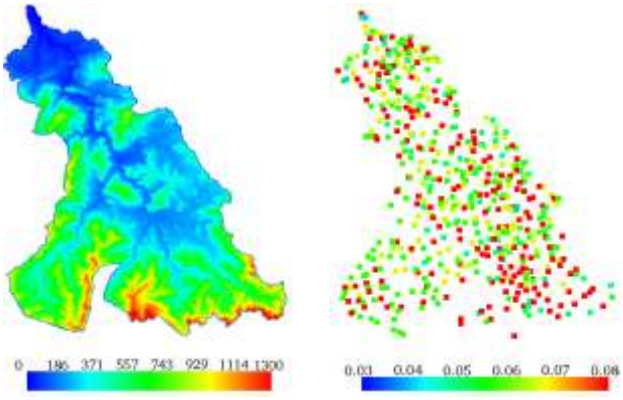


Figure 10: DEM of the study area and *Manning's* roughness coefficient

One has deliberately enclosed the domain by a rectangular shape and meshed it by finite elements in order to compare our simulations CPU times with the LBM method which is based on the finite difference method. This leads to a finite elements mesh of 1,072,000 nodes. The bottom friction of the river is set using *Manning's* roughness coefficient. Figure 10 shows the spatial distribution of this coefficient which will be interpolated over the entire mesh. Figure 11 show a zoom of the mesh used in this study with a refinement of it at the minor bed.

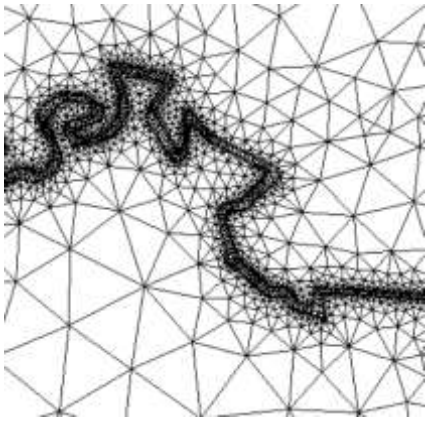


Figure 11: Zoom of the finite element mesh

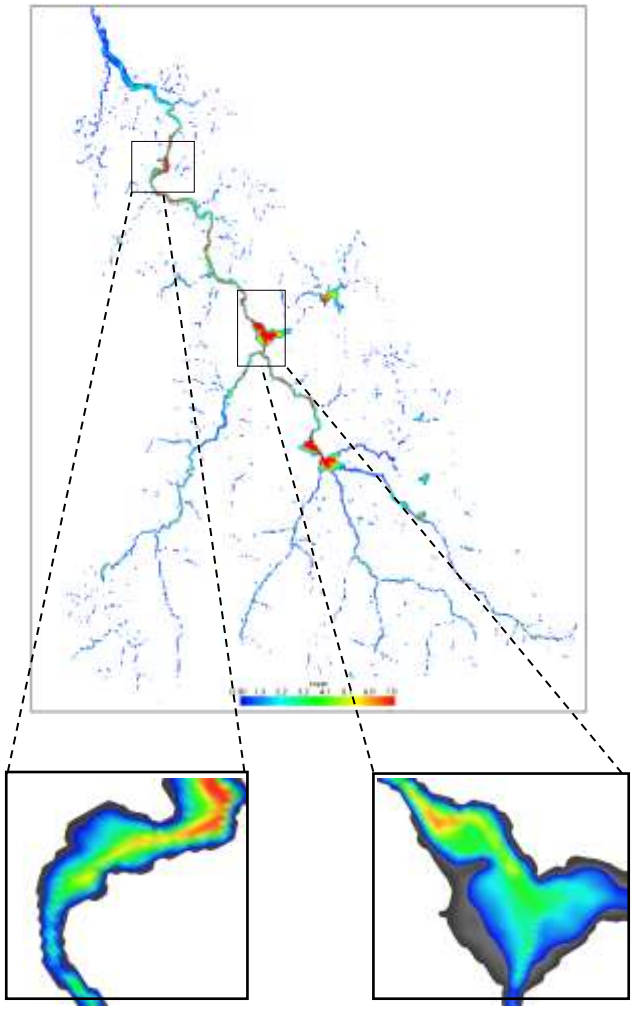


Figure 12: Simulation results show the flood at its maximum for validation rain precipitation sample and zoom at *Ossess* and *Cambo-Les-Bains* (t=24 hours against maximum flood)

One imposes "*Inflow*" boundary condition at the upstream and "*free outflow*" at the downstream of the domain. The time step used to simulate the 50 rain precipitation scenarios of three days each is equal to $\Delta t = 1.0$ second. The CPU time spent for simulating one scenario (3 days or 72 hours) is equal to 15 hours on an Intel Core i7 at 2.9 Ghz and 16 Go RAM, which leads to a total simulation time for all the scenarios equal to 31 days. To calibrate and validate the hydrodynamic model, one uses the historic event of rainfall from 03 July 2014 to 06 July 2014 (figure 9). The measured depth water was read at the two hydrological stations: *Ossess* and *Cambo-Les-Bains* (figure 13).

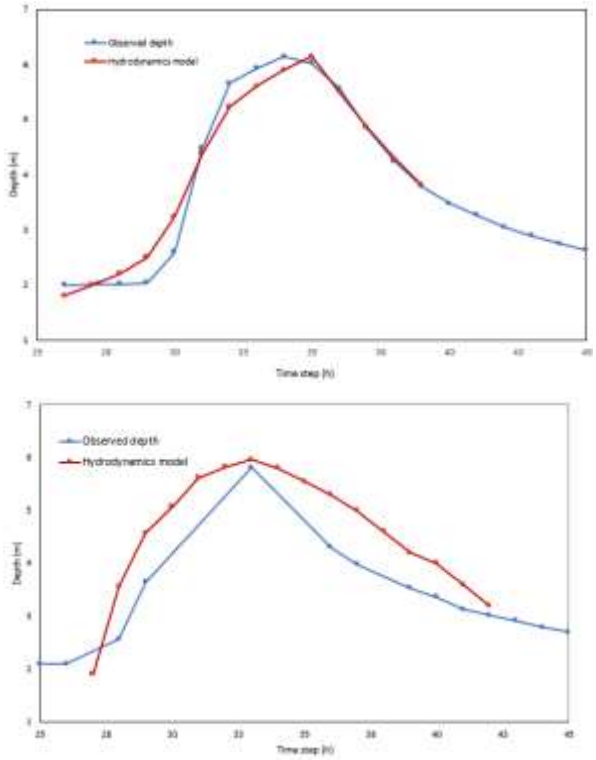


Figure 13: Comparing the time-series of water depths read at the two hydrological stations against our hydrodynamics model

The performance of the hydrodynamic model is reasonably good with a DEM grid equal to 30 m. The mismatch is mainly due to the coarsening of the grid. Notice that the objective in this study is to capture the dynamics of the flow and the extend of flood and also to obtain the maximum water level reached for a given scenario.

The proposed MH-CNN inundation modelling framework is explained in figure 14.

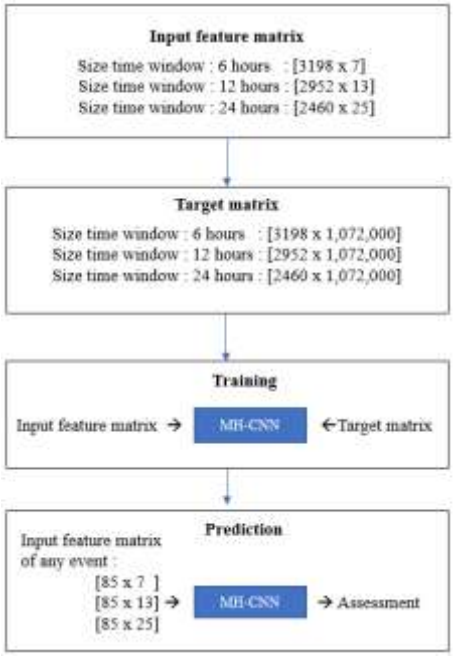


Figure 14: The proposed MH-CNN inundation modelling framework with different size time window

Several values of the size time window (antecedent or previous time-steps) are tested for 6, 12 and 24 hours (figure 15). The objective is to determine the optimal value of the size time window to obtain the longest period predicted (number of hours-ahead) for the water heights with good precision. This leads to a total of 7, 13 and 25 input variables. The input and target variables are defined by converting the fifteen rain precipitation curves applied on the domain and the corresponding depths predicted by the hydrodynamics model into matrices. This lead an input feature matrix of 3198x7, 2952x13 and 2460x25 respectively for 6, 12 and 24 hours size time window. The target matrix is created by converting the sequential water depth into arrays. This results in a matrix of size [3198×1,072,000], [2952×1,072,000] and [2460×1,072,000] respectively where 1,072,000 is the total number of finite element nodes in the domain.

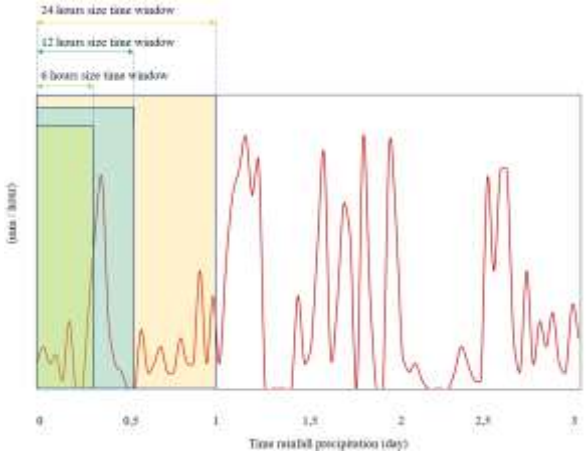


Figure 15: Example of the different size time window proposed (6h, 12h and 24h)

Fifty rain events were used for testing the model with the duration of 3 days each. The hydrodynamics model is then run fifty times using those historic and synthetic rain precipitations to produce different flood conditions in the study site described by DEM. These samples are divided into 48 samples for the training set, one sample for validation and one for the test set. This division was made manually to obtain a good representation of strong and weak rain events in all sets. The sample for validation corresponds to the 03/07/2014 rain precipitation and the sample for test corresponds to an arbitrary synthetic rain precipitation (figure 16). For each calculation, one varies the size time window to 6, 12 and 24 hours.

During the running process of the CNN and MH-CNN, different regularization techniques, including 'early stopping' (stops the training process when the model performance does not improve after a certain number of iterations), and 'dropout' are applied to prevent the model from overfitting.

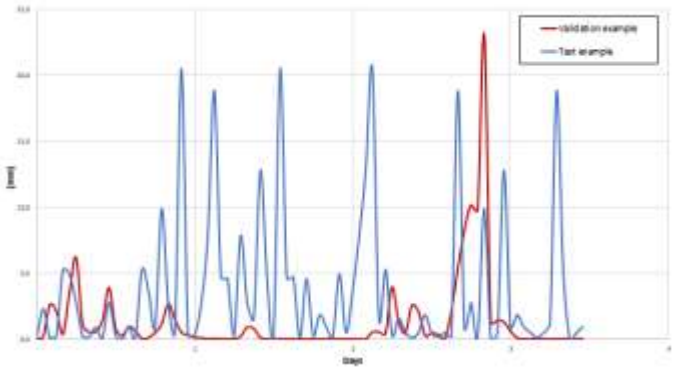
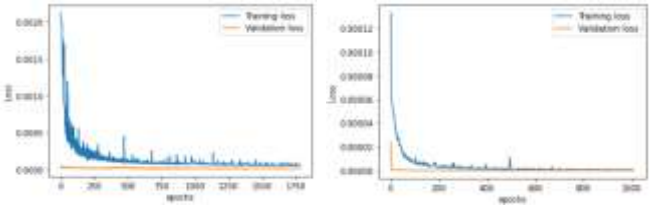


Figure 16: Test and validation rain precipitation samples

To validate the model, for example, for 24 hours size time window, the CNN model run at around 0.6 s per epoch, and the total time spent on the model training was shown to be about 18 minutes. The MH-CNN model run at around 1.6 s per epoch, and the total time spent on the model training was shown to be about 26 minutes. Figure 18 shows the two models training results. Figure 18a shows loss and validation loss for CNN, and figure 18b shows loss and validation loss for MH-CNN. When the model was stably trained, the change in loss and validation loss were similar to an exponential function with base less than one. The training loss initially exhibited unstable changes; however, it gradually decreased as the epochs progressed, indicating that the model training and generalization were stably performed.



(**a**) (**b**)

Figure 18: (**a**) Result of the CNN training; (**b**) Result of the MH-CNN training

To test our approach, one compares the results with those obtained by the hydrodynamic model at two points (*Ossess* and *Cambo-Les-Bains* stations) where the flooding reached its peak. The test rain precipitation is the blue curve in (figure 16). The computation time for whole test for one rain event (3 days) is in the range from 3 to 5 seconds. The computation time with the physically based numerical model was on average equal to 15 hours. The results are compared for size time window of 6, 12 and 24 hours. The maximal depth reached at *Ossess* station is about 4.5 meters and 5.5 meters at *Cambo-Les-Bains* station.

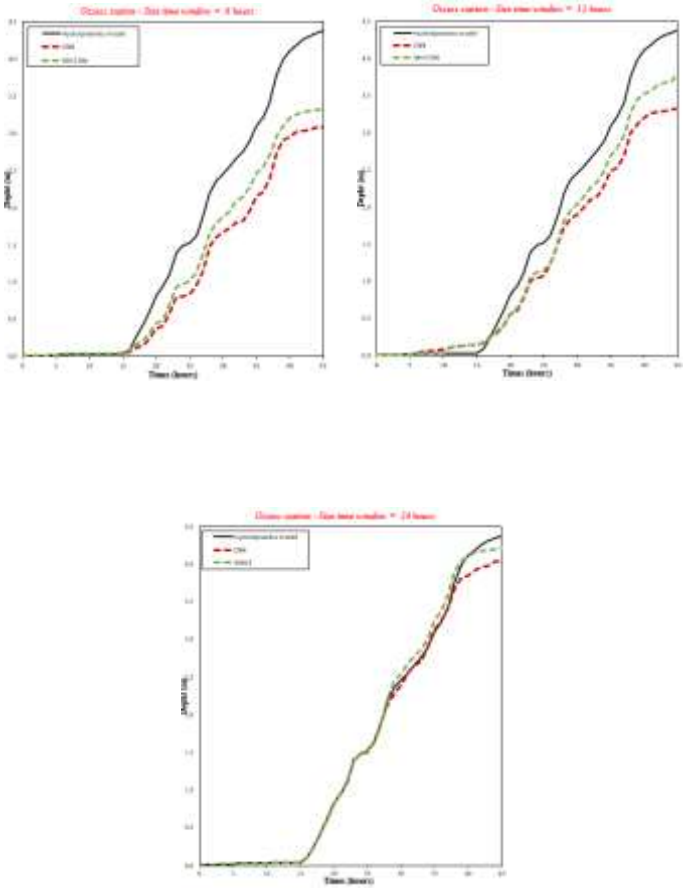


Figure 19: *Osses* station: Prediction with different size time window

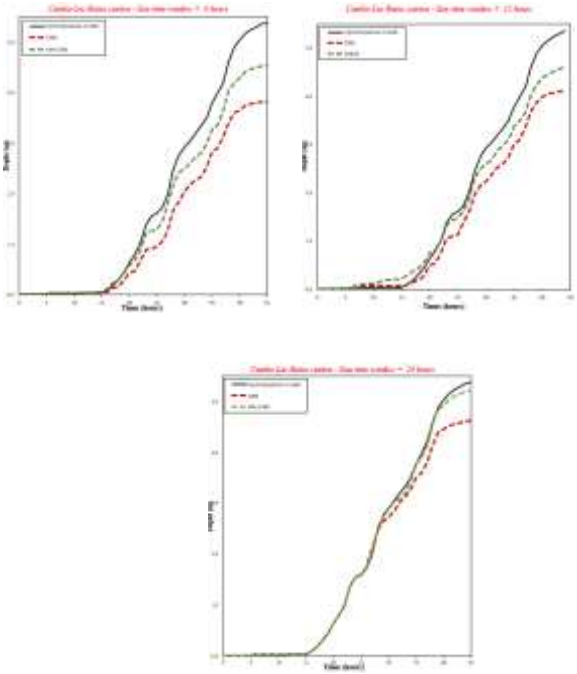


Figure 20: *Cambo-Les-Bains* station: Prediction with different size time window

To demonstrate the importance of the size time window on the quality of water height predictions, the following figures show the number of hours predicted by MH-CNN method for each time size at the *Ossess* station. Notice that the same results were obtained at *Cambo-Les-Bains* station. Increasing the size time window improves significantly the prediction in terms of number of hours-ahead predicted. Indeed, one see that for size time window equal to 24 hours, the prediction is exact with four-hours-ahead while for time size equal to 12 hours, the prediction is exact with two-hours-ahead and it doesn't exceed one-hour-ahead for a time size equal to 6 hours.

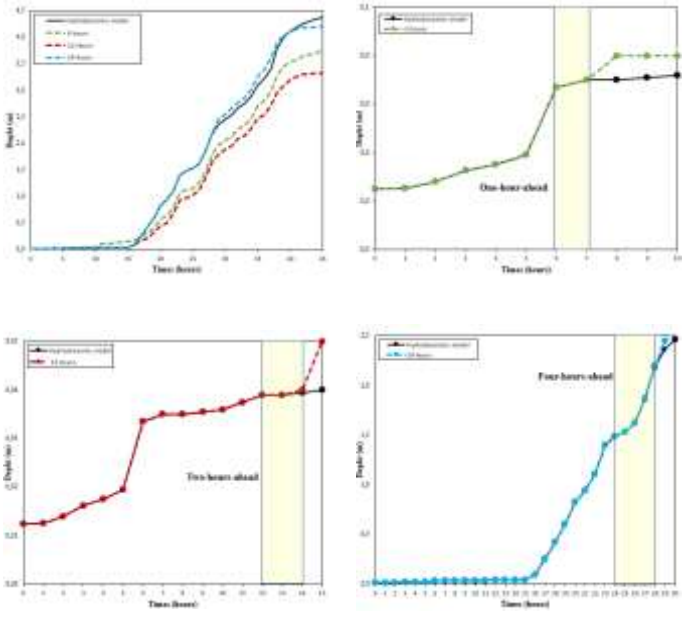


Figure 21: MH-CNN for hours prediction ahead at *Ossess* station

The test data set described above was used to evaluate the model's performance. Figure 22 shows the dispersion of the model's predicted and actual values according to 3 days simulations results. R^2, which indicates the relationship between the actual and predicted value, was relatively high at 0.99. The predicted values showed a similar trend to the actual values overall. Table 2 shows the R^2, NSE, and RMSE to evaluate the model for 6, 12 and 24 hours size time window.

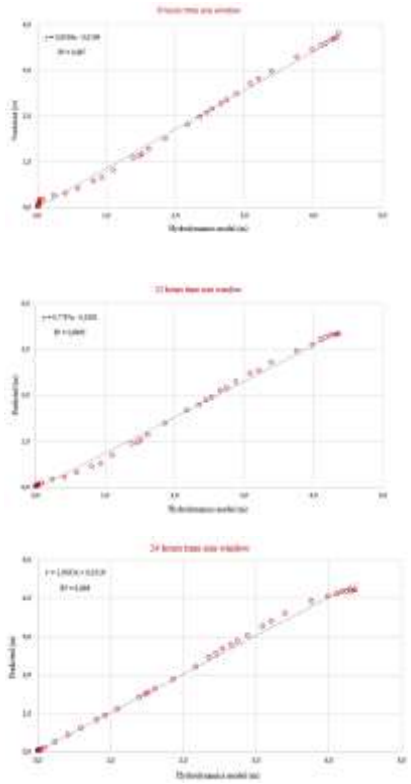


Figure 22: Scatter plot of predicted and actual values of depth water at *Ossess* station

The values were more 0.99, indicating a very high correlation between the predicted and actual values. For NSE, values of 0.60 - 0.80 are generally judged as moderate to good and values exceeding 0.80 as good. As NSE in this model was greater than 0.90, its performance could be judged as good for all size time window. RMSE were 0.57 m, 0.44 m and 0.12 m for 6, 12 and 24 hours size time window respectively indicating higher model performance for the last one.

Table 2: Score for MH-CNN at *Ossess* station

Contents	R^2	NSE	RMSE (m)
6 hours	0.9978	0.83	0.57
12 hours	0.9972	0.87	0.44
24 hours	0.9987	0.92	0.12

7. Conclusion

A Multi-headed CNN based ensemble approach for real-time water level prediction in large-scale areas was implemented and tested in large floodplain of 870 km^2 area. The model is based on feedforward neural network and trained with only precipitation rates as input and 2D distributed water depth levels as output. The model was tested with test events and compared with the physically based numerical model. The Multi-headed CNN model has the capability to make predictions for a domain consisting of more than millions of finite element nodes and it is shown that size time window improves hours-ahead predictions.

The developed model achieves computation times and accuracies that can be considered as sufficient for real-time forecasts and can be seen as a step towards 2D real-time flood prediction for large-scale areas.
If the minimum length of the input data is obtained, the use of MH-CNN compared to the results calculated by the traditional CFD models is evaluated as an important achievement as it can effectively predict the flow rates quickly and accurately.

References

[1] S. Hadji, 'Méthode de résolution pour les fluides incompressibles', Ph.D. thesis, Compiegne University of Technology, 1995.

[2] S. Hadji, G. Dhatt, Asymptotic-Newton method for solving incompressible flows, International Journal for Numerical Methods in Fluids, vol. 25, 861-878, 1997.

[3] B. Chopard, J.L. Falcone J. Latt, The Lattice Boltzmann advection diffusion model revisited, The European Physical Journal - Special Topics,Vol. 171, pp. 245-249, 2009.

[4] Nvidia, C. U. D. A. Nvidia cuda c programming guide. NVIDIA Corporation, 120, 18.8, 2011.

[5] Syed Kabir, Sandhya Patidar, Xilin Xia, Qiuhua Liang, Jeffrey Neal, Gareth Pender, A deep convolutional neural network model for rapid prediction of fluvial flood inundation, Journal of Hydrology, 590, 2020.

[6] Kidoo Park, Younghun Jung, Kyungtak Kim, Seung Kook Park, Determination of Deep Learning Model and Optimum Length of Training Data in the River with Large Fluctuations in Flow Rates, Water, 2020.

[7] Simon Berkhahna, Lothar Fuchsb, Insa Neuweilera, An ensemble neural network model for real-time prediction of urban floods, Journal of Hydrology, 2019.

[8] Kiranyaz, S., Avci, O., Abdeljaber, O., Ince, T., Gabbouj, M., Inman, D.J. 1D Convolutional Neural Networks and Applications., Mechanical Systems and Signal Processing, Volume 151, 2021.

[9] Zihlmann, M., Perekrestenko, D., Tschannen, M., Convolutional Recurrent Neural Networks for Electrocardiogram Classification. Retrieved from http://arxiv.org/abs/ 1710.06122, 2017.

[10] Kiranyaz, S., Ince, T., Hamila, R., Gabbouj, M., Convolutional Neural Networks for patient-specific ECG classification. Conf. Proc. ... Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. IEEE Eng. Med. Biol. Soc. Annu. Conf. 2015, 2608–2611, 2015.

[11] Abdeljaber, O., Avci, O., Kiranyaz, M. S., Boashash, B., Sodano, H., Inman, D. J., 2018. 1-D CNNs for structural damage detection: Verification on a structural health monitoring benchmark data. Neurocomputing 275, 2018.

[12] SRTM DEM Shuttle Radar Topography Mission, http://www.viewfinderpanoramas.org/Coverage map viewfinderpanoramas_org3.htm

Author Profile

Sofiane Hadji Scientific Director in Civil and Environmental Technology at SIXENSE Engineering (Vinci Group) in France since 2010. Dr. Hadji received in 1995 his Ph.D. degree in Mechanical engineering (Modeling & Optimization of Products & Structures) and his M.Sc.Eng. in 1991 from the Compiègne University of Technology. He also obtained in 1990 his B.Eng. in Civil Engineering from the Algiers University of Technology (Algeria). His research interests include Computational civil engineering, Hydrodynamics modeling, Optimization methods. To date, he authored more than 60 technical reports and 20 international journal papers.