Project Report : CS 7643

Chanyu Yang
Georgia Tech
172 4th St NW, Atlanta
cyang490@gatech.edu

Kai Kleinbard
Georgia Tech
172 4th St NW, Atlanta
kkleinbard3@gatech.edu

Abstract

Flooding can have significant impacts on people, communities, properties, critical infrastructures, etc. Hydrodynamic models provide flood risk management with cost effective tools but are also computationally expensive. This project tried to investigate the feasibility of using deep learning models as surrogate models for flood inundation modeling. The convolutional neural network (CNN) was trained using the outputs from a two-dimensional hydrodynamic model (LISFLOOD-FP) to predict flood inundation depths. Then the pretrained CNN was applied on two historical events January 2010 and October 2011 to simulate flood inundation. The CNN was able to capture flooded pixels with decent accuracy as indicated by some quantitative assessment. The estimated error was below 0.04 for all batches. The proposed CNN model proves the feasibility of deep learning models for surrogate flood inundation modeling.

1. Introduction

Flood inundation forecasting is useful for flood risk management so as to reduce the threats to people, communities, properties, critical infrastructures, etc. As a component of flood risk management, the hydrology/hydrodynamic models are one of the commonly used methods for 2D flood inundation modeling. These methods are normally accurate but computationally expensive and therefore not suitable for applications where a large number of model runs are required [1]. The use of deep (machine) learning is one of the possible solutions that can closely emulate the outputs of 2D hydrodynamic models [2]. This project aims to develop a surrogate model using deep learning for flood inundation forecasting that can alleviate the computational burden while maintaining accuracy. More specifically, we want to investigate the feasibility of using deep learning for 2D flood water depths forecasting in a fluvial flood setting.

1.1 Related Research

Flood inundation forecasting is useful for flood risk management so as to reduce the threats to people, communities, properties, critical infrastructures, etc. The hydrology/hydrodynamic models are one of the commonly used methods for 2D flood inundation modeling, which are normally accurate but computationally expensive. The use of deep (machine) learning is one of the possible solutions that can closely emulate the outputs of 2D hydrodynamic models. In our research, we have honed in on two other methods, one uses GANs and another uses an RNN with an encoder and decoder. The challenge is to take sparse images of possible flooding and predict whether a flood is actually happening. In the generative deep learning approach, the researchers [12] present two models: GAN-based and VAE-GAN-based, to increase the resolution of precipitation forecast data by a factor of 10 and calibrate the forecast to better make predictions about rainfall. In the paper, the generator in the VAE-GAN (Variational Auto-Encoder) takes in a noise input (of sparse image data of precipitation) and attempts to generate high-resolution rainfall data. The discriminator is then trained to distinguish between the high-resolution predictions from the generator and the corresponding true high-resolution rainfall data.

Finally, an additional approach [13]uses a stacked autoencoder (SAE) with a recurrent neural network (RNN). The SAE-RNN model uses SAE to compress (encode) high dimensional data into a lower dimensional latent space,

with flood features. It uses "an RNN to forecast multistep-ahead flood features based on regional rainfall patterns", then it uses SAE to reconstruct (decode) the multistep-ahead forecasts of flood features to predict the flood depths of the area map.

1.2 Limits of Current Approaches

Flood predictions require multiple types of data that are hard to come by. For example, amount of rainfall occurring on a real-time basis, the rate of change in river stage, duration, intensity and areal extent, and knowledge about the characteristics of a river's drainage basin, such as soil-moisture conditions, ground temperature, snowpack, topography, vegetation cover, and impermeable land area. In addition, inundation simulation is now constrained by the quality of boundary conditions and flood defenses. Because of the massive amount of data, the timely nature and complexity of it, flood predictions are historically inaccurate [12].

Successfully being able to predict flooding extent will allow developers, governments and citizens to understand how to prepare better for flood events. This is becoming increasingly important as global warming accelerates. Many lives are at risk without the ability to correctly predict flooding. In addition, the ability to model flood extent ensures that emergency personnel can better prepare for weather changes.

In the following sections, details about data, methodology and results will be introduced in section 2, 3 and 4 respectively.

2. Data

The flood inundation modeling means the floods are modeled in a two-dimensional way. The resolution of the model mainly depends on the resolution of the remote sensing data used (terrain data in this project). Since this study is at the early stage, and trying to answer how suitable it would be to use deep learning models as a type of surrogate models for flood modeling, we built our model in a coarser resolution (30m) instead of the original resolution (2m). For more information about preprocessing from 2m to 30m, please refer to section 2.3. The preparation processes of inputs will be introduced from section 2.2 to 2.4 while preprocessing target data will be discussed in section 2.5.

2.1 Study Location

Based on the data quality and availability, the catchment of river Dodder (from Waldron's Bridge to the Ballsbridge) and its tributary Slang (Dundrum) were selected as the initial study area.
Data sources are Office of Public Works (OPW) (for river information, terrain data and hydrometric data), and Google Earth (for river information).

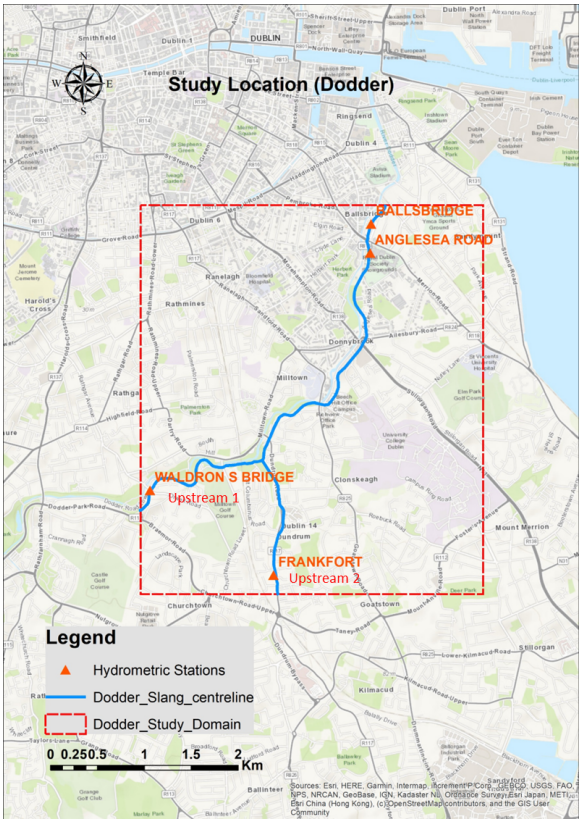

Figure 1. Study Location - Dodder

2.2 River information

The river information includes river location, river channel widths and bank heights.
The channel widths were collected from the Office of Public Works (Ireland) which are cross-section survey data, and from Google Earth that the widths were delineated from the satellite image. Shown in Figure 2 is an example of a cross section. In the "PROFILE" part, #1, #2 and #4 are the left bank, center of the channel and right bank

respectively. Three columns showing from left to right are offsets, elevations and manning coefficients respectively. For the reason that data from cross section survey didn't fully cover the river reach of our study domain, using satellite image from Google Earth to delineate rest of the cross sections was considered as the alternative for supplementing channel widths data. Shown in Figure 3 is an example of delineating channel width from Google Earth satellite image. After channel width data were collected from both sources, these data were combined. Furthermore, in accordance with the model resolution, the channel widths were processed from vector data to raster data using Kriging interpolation method. The combination and interpolation were processed using ArcGIS 10.7.

The collection of river location and bank heights will be introduced in section 2.3 Terrain Data.

```
A
Dodder_main Channel
19589.165
COORDINATES
1 317952.09    234181.73
FLOW DIRECTION
0
DATUM
0
RADIUS TYPE
2
DIVIDE X-SECTION
0
SECTION ID
D-001
INTERPOLATED
0
ANGLE
004 44 21
PROFILE          21
    -48.95      3.18    0.03
    -47.77      3.16    0.03<#1>
    -46.85     -1.84    0.03
      1.25     -1.69    0.03<#2>
      3.95     -1.64    0.03
      4.07     -1.63    0.03
     11.90     -1.53    0.03
     18.95     -1.48    0.03
     18.98     -1.50    0.03
     19.33     -1.47    0.03
     29.48     -1.53    0.03
     43.71     -2.03    0.03
     43.77      2.70    0.03<#4>
     44.38      2.74    0.03
     46.59      2.78    0.03
     47.28      2.77    0.03
     47.62      2.71    0.03
     48.89      2.65    0.03
     49.22      2.78    0.03
     49.23      2.80    0.03
     49.25      2.78    0.03
```

Figure 2. Example of cross section survey

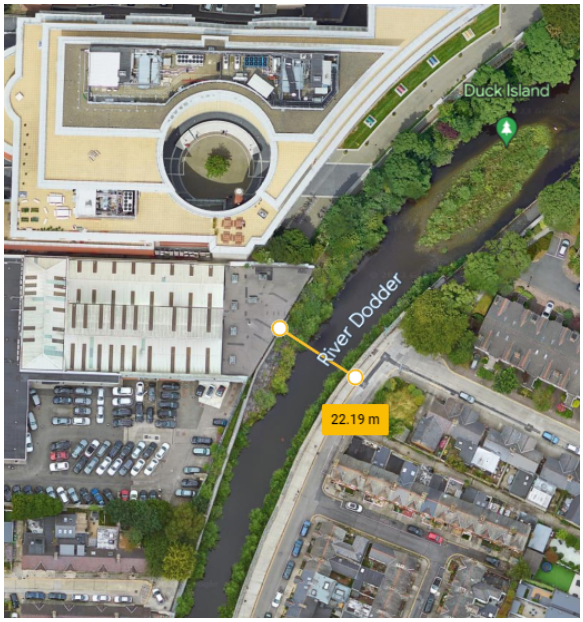

Figure 3. Example showing how to delineate channel width from Google Earth

2.3 Terrain Data (Remote Sensing Data)

The terrain data used in this project is the Digital Terrain Model (DTM), which is 2m resolution Light Detection and Ranging (Lidar) data and can be downloaded from the Open Topographic Data Viewer [5]. The data downloaded is called "LIDAR DSM Hillshade OPW NASC 2m Ireland (ROI) ITM MH TIFF" which covers the study area (Dodder).

The original DTM (resolution: 2m) were then preprocessed into 30m resolution using cubic convolution resampling method by ArcGIS 10.7.

The river location is shown as the centreline of the river channel. This can be delineated from the DTM.

For the river bank heights, we assumed that they are the same as the DTM.

2.4 Hydrometric Data

The data can be downloaded from EPA Ireland HydroNet data portal [6]. Flow records (discharge) needed for this project are from hydrometric station Waldron's Bridge (09010, upstream gauge 1) and Frankfort (09011, upstream gauge 2). Figure 1 shows the locations of the two upstream stations. The temporal resolution of these records is 15 mins.

2.5 Hydrodynamic Model

There are different ways to obtain flood inundation extent as target data, for instance, historical records from remote sensing images, in-situ measurements, and modeling approach. In this project, a model called LISFLOOD-FP was used for modeling flood inundation extent using the aforementioned information (section 2.1 to 2.4) as inputs. LISFLOOD-FP is a two-dimensional raster-based hydrodynamic model specifically designed to simulate floodplain inundation. The model simulates the propagation of flood waves along channels and across floodplains using simplifications of the shallow water equations [7]. Flood water depths were generated after running the LISFLOOD-FP.

2.6 Preparing Samples for Training and Testing

Four flood events were picked based on the flow records of the upstream station 1 (Waldron's Bridge). These flood events occurred during the years 2010, 2011, 2014, and 2018.

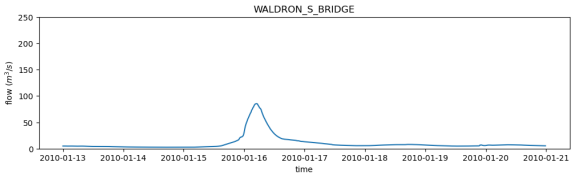

Figure 4. Flood event 2010

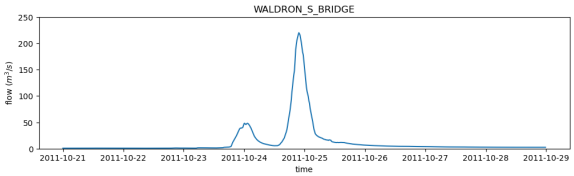

Figure 5. Flood event 2011

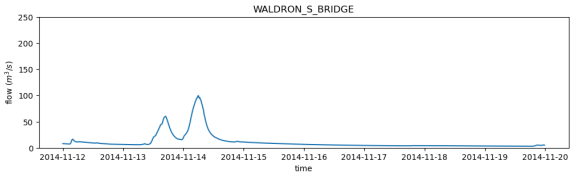

Figure 6. Flood event 2014

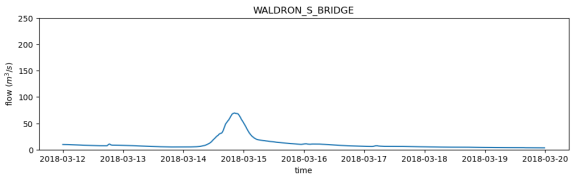

Figure 7. Flood event 2018

The events 2014 and 2018 were used for training while the events 2010 and 2011 were for testing. To cover more flooding scenarios, flow records of the events 2014 and 2018 were further preprocessed based on the method in the paper [4] (see equation 1), to produce synthetic hydrographs. But instead of customizing peak flows when generating synthetic hydrographs, this process was simplified by using a scaling factor (SF).

$$Q_{syn} = Q_{obs} \times \frac{Peak_{max}}{Q_{max}} \; where \; Peak_{max} > Q_{max} \quad (1)$$

Q_{syn} is the synthetic flow, Q_{obs} is the observed flow, $Peak_{max}$ is the peak flow that user can customize, Q_{max} is the observed maximum flow record. We simplified this equation into the following format.

$$Q_{syn} = Q_{obs} \times SF \quad (2)$$

Details of data for training and testing were shown in Figure 8 and 9, and table 1.

Data structure of training and testing data were shown in figure 10 to figure 13. Note that different from hydrodynamic modeling where flow records from both upstream gauges needed to be considered, only the data from upstream gauge 1 was used as the input of the deep learning model. Because upstream gauge 1 is located in the main river reach with maximum flow rate around 250 m^3/s while maximum flow rate of gauge 2 is below 20 m^3/s. We hereafter simplified the input matrix by excluding flow records from upstream gauge 2. The input matrix for training contained 3040 samples with 10 features per sample. These samples were associated with hydrographs A to D. The current time step, flow records from the current

time step and from the previous eight time steps were composed as the feature vector for each sample. More specifically, the reason for choosing the antecedent 8 time steps as part of the feature vector was that this associated with 8 * 15 mins=120 mins=2 hrs of flood record which could have significant impact on flood risk assessment especially for analyzing flash floods. For every hydrograph, it contained records of 760 time steps. So the total samples = 760 records * 4 hydrographs (A-D). In terms of the target matrix, the sample size was the same as the input matrix except the feature vector contained water depths of all the pixels within the study domain. The data structure of testing data was similar to that of training data.

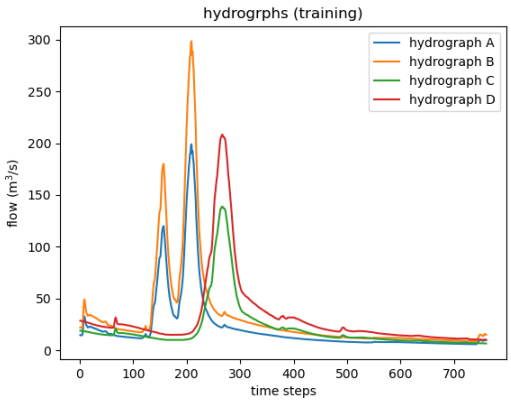

Figure 8. Hydrographs for training

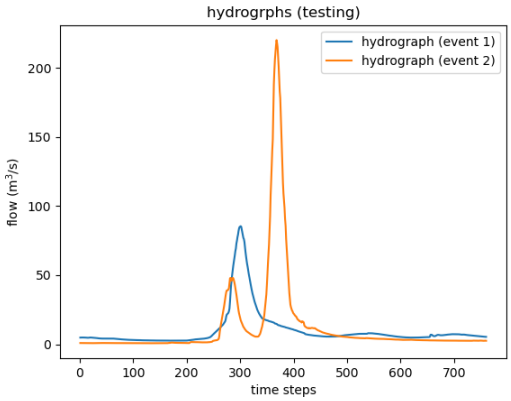

Figure 9. Hydrographs for testing

Hydrograph	Time start	Time end	SF	purpose
A	2014-11-12	2014-11-19	2	training
B	2014-11-12	2014-11-19	3	training
C	2018-03-12	2018-03-19	2	training
D	2018-03-12	2018-03-19	3	training
Event 1	2010-01-13	2010-01-20	-	testing
Event 2	2011-10-21	2011-10-28	-	testing

Table 1. Details of synthetic hydrographs for training

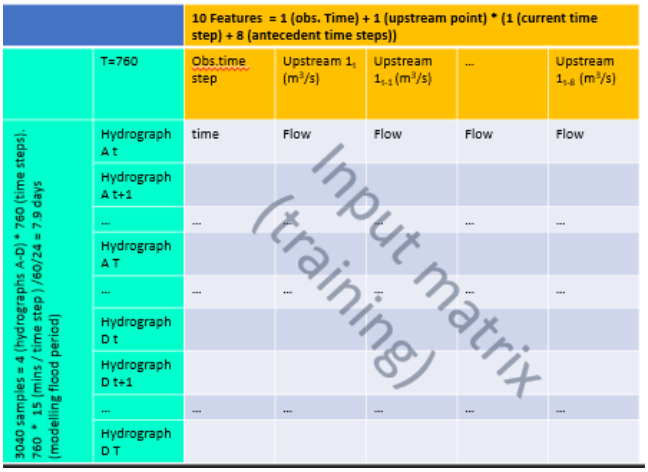

Figure 10. Data structure of input matrix (training)

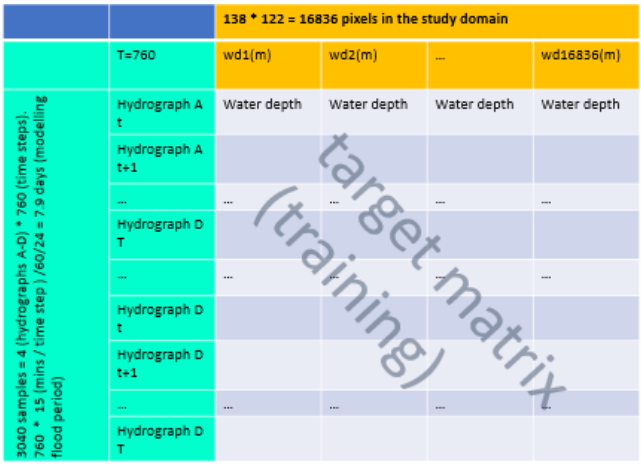

Figure 11. Data structure of target matrix (training)

		10 Features = 1 (obs. Time) + 1 (upstream points) * (1 (current time step) + 8 (antecedent time steps))				
	T=760	Obs.time step	Upstream 1_t (m^3/s)	Upstream 1_{t-1} (m^3/s)	...	Upstream 1_{t-8} (m^3/s)
1520 samples = 2 (hydrographs event 1 & 2) * 760 (time steps). 760 * 15 (mins / time step) /60/24 = 7.9 days (modelling flood period)	Hydrograph 1 t	time	Flow	Flow	Flow	Flow
	Hydrograph 1 t+1					
	...	...	...		...	...
	Hydrograph 1 T					
	Hydrograph 2 t					
	Hydrograph 2 t+1					
	...	...	...	...	...	...
	Hydrograph 2 T					

Input matrix (testing)

Figure 12. Data structure of input matrix (testing)

		138 * 122 = 16836 pixels in the study domain			
	T=760	wd1(m)	wd2(m)	...	wd16836(m)
1520 samples = 2 (hydrographs event 1 & 2) * 760 (time steps). 760 * 15 (mins / time step) /60/24 = 7.9 days (modelling flood period)	Hydrograph 1 t	Water depth	Water depth	Water depth	Water depth
	Hydrograph 1 t+1				
	...	...	...	...	...
	Hydrograph 1 T				
	Hydrograph 2 t				
	Hydrograph 2 t+1				
	...	...	...	...	...
	Hydrograph 2 T				

target matrix (testing)

Figure 13. Data structure of target matrix (testing)

3. Model Development

After data preprocessing, we created a model that reflected the model in the original paper[4]. The model as shown in figure 10 consisted of two one dimensional convolutional layers, and three linear layers. The aim of this model was from 3040 samples of synthetic data (which represented 4 hydrographs with 760 timesteps, 4*760) and 10 features, 1 observed time, + 1 upstream point, 1 current timestep and 8 antecedent time steps, we would output a flood terrain map showing the depth of the flooding. We used data from Ireland (instead of the UK that the original paper used).

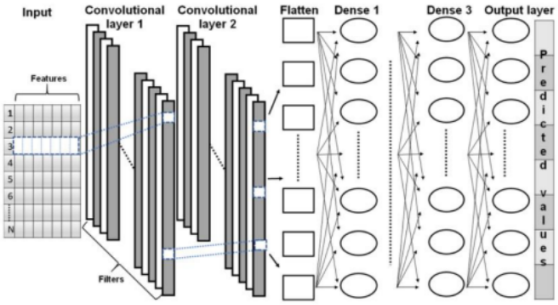

Figure 10[4]**:** the model described in the paper used two 1 dimensional convolutional layers, each was normalized using batch normalization. They were then flattened, and pushed through three linear layers.

One dimensional convolutional layers are used for temporal convolution. The layer is convolved with the input over a single temporal dimension. The benefit of a 1D convolutional layer is that it can be used for dimensionality reduction, decreasing the number of feature maps while retaining salient features. In [8] described the 1D convolutional application in mathematical terms:

For each layer of the 1D forward propagation we can use the expression:

$$x_k^l = b_k^l + \sum_{i=1}^{N_{l-1}} conv1D\left(w_{ik}^{l-1}, s_i^{l-1}\right) \qquad (1)$$

Where x_k^l is defined as the input and b_k^l is defined at the bias of the kth neuron at layer l,. The weights, w_{ik}^{l-1} is the kernel (in our case the kernel sizes were 2, with padding 0) and the output arrays are s_i^{l-1}. The intermediate output of each layer is defined as $y_k^l = f(x_k^l)$, where f is the activation function (in our case ReLU). Note we have $s_k^l = y_k^l$. We will convolve each output of the convolutional layer with the kernel of size 2 (shown as w) in (1).

Our loss function can be described as

$$E_p = \text{MSE}\left(\boldsymbol{t}^p, \left[y_1^L, \cdots, y_{N_L}^L\right]'\right) = \sum_{i=1}^{N_L} \left(y_i^L - t_i^p\right)^2 \qquad (2)$$

Where we sum the predicted from the ground truth.

During backpropagation, we obtain the gradients of the weights with respect to the error and the gradients of the output of each layer with respect to the error as:

$$\frac{\partial E}{\partial w_{ik}^{l-1}} = \Delta_k^l y_i^{l-1} \text{ and } \frac{\partial E}{\partial b_k^l} = \Delta_k^l \quad (3)$$

$$\frac{\partial E}{\partial s_k^l} = \Delta s_k^l = \sum_{i=1}^{N_{l+1}} \frac{\partial E}{\partial x_i^{l+1}} \frac{\partial x_i^{l+1}}{\partial s_k^l} = \sum_{i=1}^{N_{l+1}} \Delta_i^{l+1} w_{ki}^l \quad (4)$$

It is important to get the dimensions correct in the implementation. We need to pass in

Architecture of our Model:
ConvNet ((conv1): Conv1d(1, 31, kernel_size=(2,), stride=(1,)) (bn1): BatchNorm1d(31, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True) (conv2): Conv1d(31, 32, kernel_size=(2,), stride=(1,), padding=(1,)) (bn2): BatchNorm1d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True) (fc1): Linear(in_features=320, out_features=512, bias=True) (bn3): BatchNorm1d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True) (fc2): Linear(in_features=512, out_features=256, bias=True) (bn4): BatchNorm1d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True) (fc3): Linear(in_features=256, out_features=128, bias=True) (bn5): BatchNorm1d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True) (fc4): Linear(in_features=128, out_features=16836, bias=True) (dropout): Dropout(p=0.2, inplace=False) (criterion): MSELoss())

Figure 11: This is a summary of our model. Dimensions are important in a 1 dimensional CNN. Note that the input has a channel in equal to 1 dimension and a channel out of 31.

Figuring out the correct dimensions using a 1D CNN can be confusing (at first we thought we would use an even number such as 32). The reason we have to have an out_channel of 31 is due to the 0 padding and size = 2 kernel. This was confusing at first to use odd numbers of channels for an even sized input. To resolve this we apply the following formula [9]:

$$L_{out} = \left\lfloor \frac{L_{in} + 2 \times \text{padding} - \text{dilation} \times (\text{kernel\_size} - 1) - 1}{\text{stride}} + 1 \right\rfloor \quad (5)$$

Where we have (32 -1)/1 = 31 output layers (noting *dilation = 1*).

After each 1D convolution, batch normalization, which acts as a regulator stabilizing the training.

We use a batch size of 32 (as is often recommended a smaller batch size is helpful with batch normalization to ensure more accuracy when normalizing each batch). [10]

In our architecture we also use a learning rate of 0.001. We found that a higher learning rate led to information loss (where the output images did not reflect the actual target flood data images). Meanwhile, we also used a momentum of 0.1 which also meant slowing the pace of the learning, where the optimization algorithm will use 10% of the previous weight update.

You can find a link to our code here: https://drive.google.com/drive/folders/1OO-NEde2_OR3NPWaWlJSYEypI4LW_tTY?usp=sharing

4. Experiments and Results

This section described the experimental setup and architecture, different hyperparameters that were used and the experimental results. Overall, we used 1D CNN layers, followed by dense layers. We were able to correctly predict flood depths in line with the original paper.

4.1 Model training and testing

We measured our success by following a similar procedure to the original paper. We first measured our MSE loss and compared it to a validation set. Our loss followed a nice curve and reduced to less than 0.001 by about the 15th epoch:

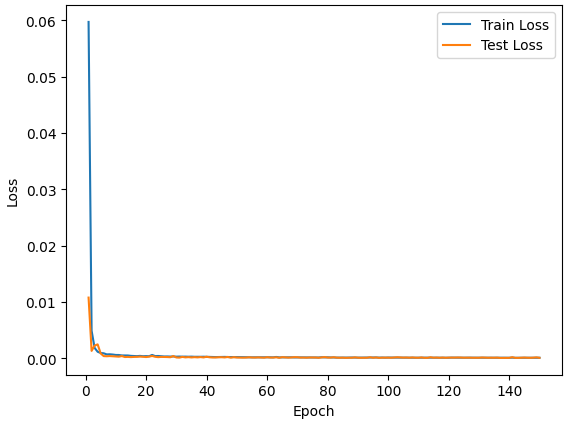

Figure 12: Using 20 epochs, our loss for training and validation are neck and neck. Loss drops quickie at around epoch 10 and then flatlines by epoch 20.

We then used a confusion matrix to identify if the output of our model would be when we inputted the test data from the actual flooding event in Ireland. The confusion matrix is displayed below:

```
Confusion Matrix:
|          | Predicted NO | Predicted YES |
|----------|--------------|---------------|
| Actual NO| 25,276,935   | 2,462         |
|Actual YES| 2,276        | 292,211       |

Accuracy: 0.9998
Precision: 0.9998
Recall: 0.9998
F1 Score: 0.9998
```

Figure 13: Confusion matrix, demonstrated high accuracy and precision.

Accuracy demonstrates that the model performs well on the dataset. Precision shows us the proportion of positive predictions that are true (the cost of false positives is high, as in a flood emergency, we would want to send emergency workers to the right locations). Recall is the proportion of true positives, and demonstrates the model did a good job showing positive outcomes.

Both the smooth loss outputs (see figure 5) and the Confusion Matrix statistics demonstrate that our model performed well based on the ground truth data.

We also, following the footsteps of the original paper, created topological graphs to identify the extent of the flooding. Using timed data, we found that our graphs matches the ground truth graphs, demonstrating that this architecture can be correctly used to predict flooding.

4.2 Hyperparameter Tuning

The hyperparameters selected were learning rate (LR) and dropout rate of the final dropout layer. Learning rate controls how much change the model in response to the estimated error each time the model parameters are updated. Dropout is a commonly used regularization technique to avoid overfitting.

Values tuned were listed in table 2. Combinations of these values were put into the model and to rebuild the model. The training input and target matrices mentioned in section 2.6 were shuffled and were splitted into training set and validation set with splitting rate 0.2. The reason to shuffle the training set is to discourage the model “remembering” the temporal change of flood inundation, for better generalization. But the testing set wasn’t shuffled so as to test how good the model could be when predicting flood inundation sequentially.

Parameter	values		
Learning rate	0.1	0.01	0.001
Dropout rate	0.1	0.2	0.3

Table 2. Hyperparameter tuning values selection

The criteria to determine the best model were based on learning curves, prediction error plots, and predicted flood depth maps.

In general, judging from the loss curves, as the learning rate increased the model learned faster but more unstable. With higher dropout rate, the model also learned slower than that with low rate.

The best model chosen was the one with LR=0.001 and dropout rate=0.2. Initially when evaluating model performance simply from learning curves, merely trivial differences could be seen from the models using LR=0.001 but with different dropout rates. Further, when estimating these models via prediction error plots, it showed that the best model surpassed its counterparts (using dropout rate=0.1 or 0.3) with lower error values and smaller variance of error across batches. Besides, when plotting the predicted flood depths back to 2D flood maps, the best model seemed to be able to predict the river channel with some floodplains along the channel. However, noisy pixels could be seen from some flood maps where some areas that are far away from the river channel were also predicted as "wet/flooded" (shown in blue). While its counterparts also had similar performance but predicted more noisy pixels. Nonetheless, with some acceptable thresholds (e.g., flood depth=0.3m), the noisy pixels could be ignored and be considered as not risky[4]. Shown from figure 14 to figure 19 are loss curves and prediction error plots of the best model and other models with same learning rate but with different dropout rate. The figure 20 and 21 are predicted flood depth maps of the best model.

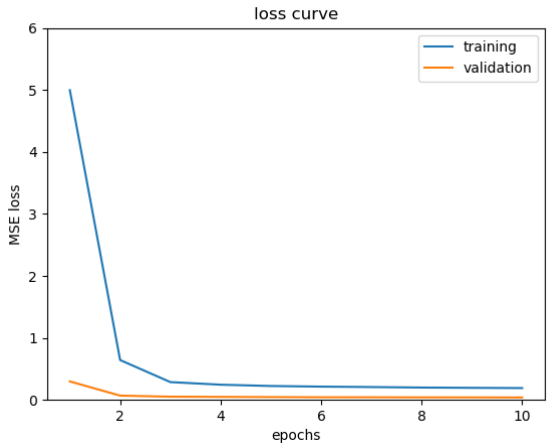

Figure 14. Loss curve of the best model (LR=0.001, dropout=0.2)

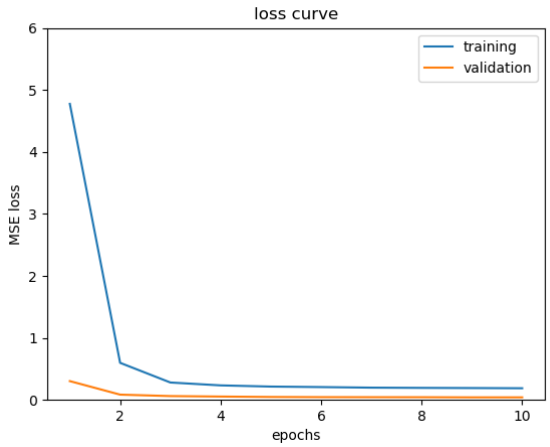

Figure 15. Loss curve (LR=0.001, dropout=0.1)

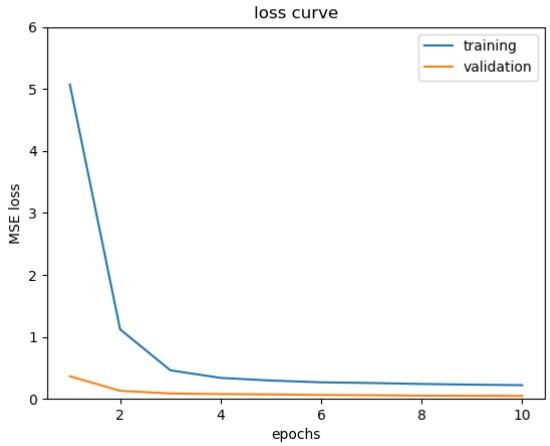

Figure 16. Loss curve (LR=0.001, dropout=0.3)

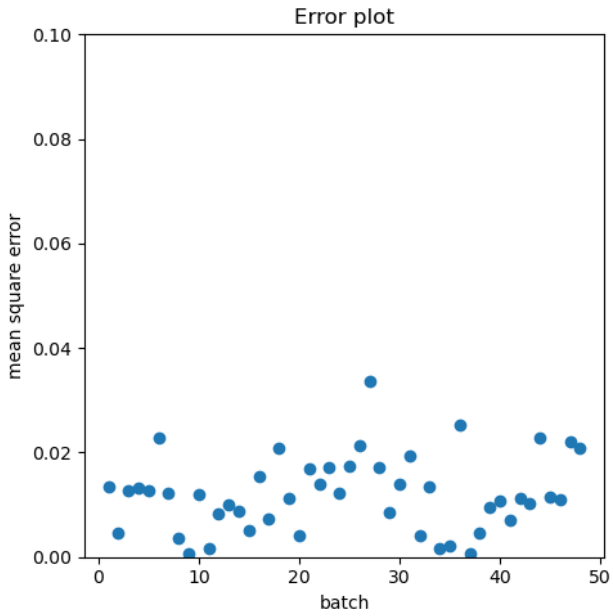

Figure 17. Prediction error of the best model (LR=0.001, dropout=0.2)

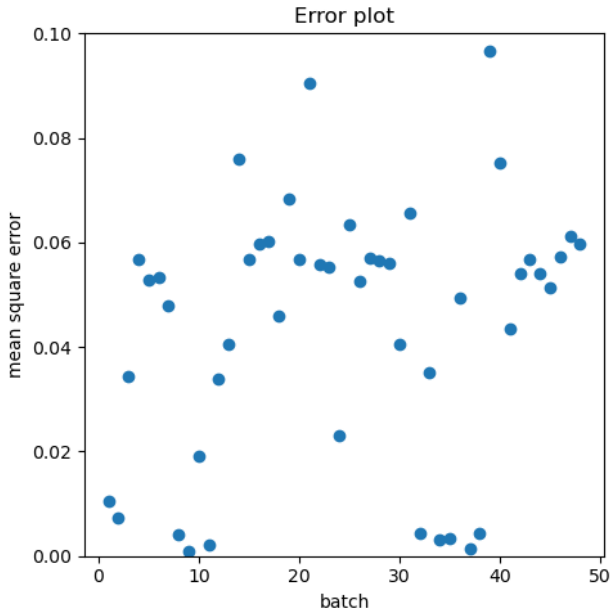

Figure 18. Prediction error (LR=0.001, dropout=0.1)

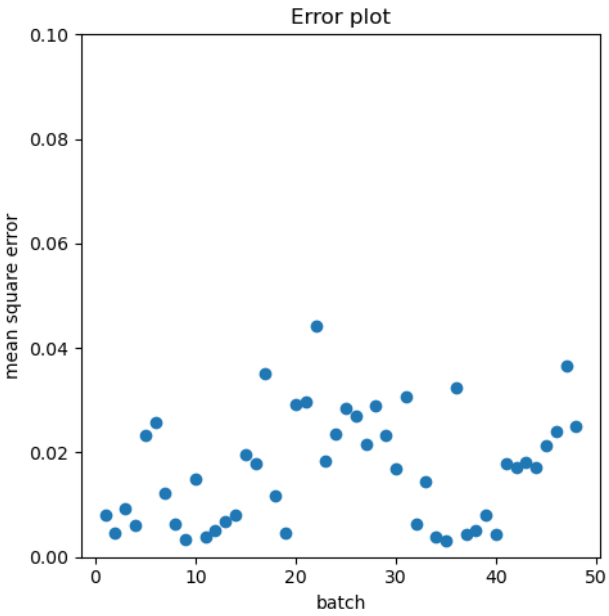

Figure 19. Prediction error (LR=0.001, dropout=0.3)

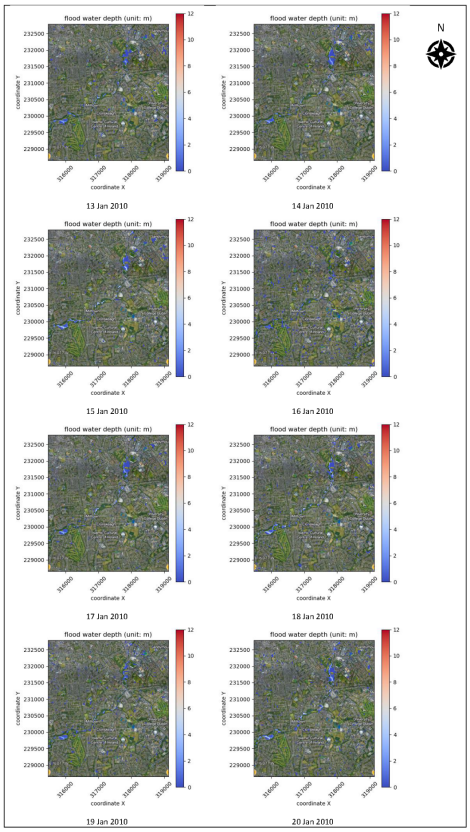

Figure 20. Flood prediction (event 1, 2010)

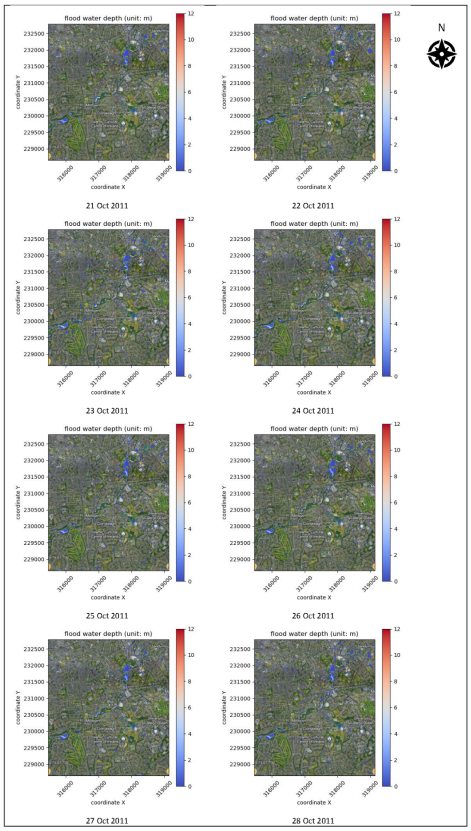

Figure 21. Flood prediction (event 2, 2011)

Conclusion and future plan

Based on the paper [4] and the flood attributes in Ireland, this project investigated the suitability and feasibility of using deep learning for two dimensional flood inundation modeling.

The model was trained and tested on a coarse resolution (30m) dataset and was able to predict flood inundation with acceptable error range.

Future research will try to focus on implementing deep learning models on higher resolutions (for instance, 2m or 5m). This could be valuable especially for highly urbanized areas. Besides, the ability of modeling temporal flood variation of the proposed model depends on the input matrix space, that is, depending on the length of the hydrographs. Future research will try to investigate other model structures (for instance, recurrent neural network) that would be more suitable and flexible for modeling flood inundation in a sequential way.

Contributions

This project was created by Chanyu Yang and Kai Kleinbard.

Yang gathered and curated the data through the EPA Ireland, the Office of Public Works (Ireland) and Geological Survey Ireland . She prepared data using hydrological, hydrodynamic and geographical approaches following the instructions from the paper [4]. Yang then preprocessed data for deep learning modeling. She helped with model architecture improvement, conducted hyperparameter tuning and visualized the data results.

Kai Kleinbard helped develop the model architecture. Kai tuned the model and trained it, helping to find the best hyperparameters. Kai analyzed the graph setting up the confusion matrix and data visualization.

Name	Contributed Aspects	Details
Chanyu Yang	Data Collection and Preprocessing. Model Architecture Revision. Hyperparameter tuning. Data Visualisation.	Collect and preprocess topological and hydrometric data for hydrodynamic model and deep learning model. Identified the inconsistency between data structure and initial model architecture, revised and improved model architecture. Hyperparameter tuning, model training, validation and testing Visualized modeling results.
Kai Kleinbard	Team project management. Data quality check. Model Architecture and Training. Data Visualisation.	Developed initial model architecture Trained model Examined/Analyzed Model success. Visualized modeling results.

Acknowledgements

This project is partly funded by the Government of Ireland Postgraduate Scholarship Programme (Irish Research Council - Met Éireann program)

References

[1] The Office of Public Works, “FLOOD RISK MANAGEMENT- Climate Change Sectoral Adaptation Plan - Prepared under the National Adaptation Framework,” 2019. https://www.gov.ie/pdf/?file=https://assets.gov.ie/46534/3575554721374f7ab6840ee11b8b066a.pdf#page=1 (accessed Sep. 11, 2022).

[2] S. Néelz and G. Pender, “Benchmarking of 2D hydraulic modelling packages,” 2010.

[3] A. J. Kalyanapu, S. Shankar, E. R. Pardyjak, D. R. Judi, and S. J. Burian, “Assessment of GPU computational enhancement to a 2D flood model,” *Environ. Model. Softw.*, vol. 26, no. 8, pp. 1009–1016, 2011.

[4] S. Kabir, S. Patidar, X. Xia, Q. Liang, J. Neal, and G. Pender, “A deep convolutional neural network model for rapid prediction of fluvial flood inundation,” *J. Hydrol.*, vol. 590, p. 125481, Nov. 2020, doi: 10.1016/j.jhydrol.2020.125481.

[5] Geological Survey Ireland, “Open Topographic Data Viewer.” [Online]. Available: https://dcenr.maps.arcgis.com/apps/webappviewer/index.html?id=b7c4b0e763964070ad69bf8c1572c9f5

[6] Environmental Protection Agency, Ireland, “EPA HydroNet.” [Online]. Available: https://epawebapp.epa.ie/hydronet/#Water%20Levels

[7] P. Bates, M. Trigg, J. Neal, and A. Dabrowa, “LISFLOOD-FP User manual.” Nov. 25, 2013. [Online]. Available: https://www.bristol.ac.uk/media-library/sites/geography/migrated/documents/lisflood-manual-v5.9.6.pdf

[8] Kiranyaz, S., Avci, O., Abdeljaber, O., Ince, T., Gabbouj, M., & Inman, D. J. (2021). 1D convolutional neural networks and applications: A survey. Mechanical Systems and Signal Processing, 151, 107398. https://doi.org/10.1016/j.ymssp.2020.107398

[9] PyTorch. (2023). Conv1d — PyTorch 1.10.0 documentation. Retrieved July 26, 2023, from https://pytorch.org/docs/stable/generated/torch.nn.Conv1d.html

[10] Wikipedia contributors. (2021, December 17). Batch normalization. In Wikipedia, The Free Encyclopedia. Retrieved 20:00, December 21, 2021, from https://en.wikipedia.org/w/index.php?title=Batch_normalization&oldid=1059993869

[11] McMillan, H., Krueger, T. & Freer, J. Hydrol. Process. 26, 4078–4111 (2012).

[12] Harris, Lucy, et al. "A generative deep learning approach to stochastic downscaling of precipitation forecasts." Journal of Advances in Modeling Earth Systems 14.10 (2022): e2022MS003120.

[13] Kao, I-Feng, et al. "Fusing stacked autoencoder and long short-term memory for regional multistep-ahead flood inundation forecasts." Journal of Hydrology 598 (2021): 126371.