**Bathymetry Inversion Using a Deep-Learning-Based Surrogate for Shallow Water Equations Solvers**

Xiaofeng Liu1,2, Yalan Song1, and Chaopeng Shen1

1Department of Civil and Environmental Engineering, Pennsylvania State University, University Park, PA, USA, 2Institute of Computational and Data Sciences, Pennsylvania State University, University Park, PA, USA

**Abstract** River bathymetry is critical for many aspects of water resources management. We propose and demonstrate a bathymetry inversion method using a deep-learning-based surrogate for shallow water equations solvers. The surrogate uses the convolutional autoencoder with a shared-encoder, separate-decoder architecture. It encodes the input equations and decodes to separate outputs for flow field variables. Utilizing the differentiability of the surrogate, a gradient-based optimizer is used to perform bathymetry inversion. Two physically based constraints on the ranges of bed elevation and slope have to be added as inversion loss regularizations to contract the solution space. Using the “L-curve” criterion, a heuristic approach is proposed to determine the regularization parameters. Both the surrogate model and the inversion algorithm show good performance. The bathymetry inversion progresses in two distinctive stages, which resembles the sculptural process of initial broad-brush calving and final fine detailing. The inversion loss due to flow prediction error and the two regularizations play dominant roles in the initial and final stages, respectively. The surrogate architecture (whether with both velocity and water surface elevation or velocity only as outputs) does not have significant impact on inversion result. The methodology proposed in this work, an example of differentiable parameter learning, can be similarly used in the inversion of other important distributed parameters such as roughness coefficient.

1. **Introduction**

River bathymetry is critical for many aspects of water resources management, for example, flood inundation mapping, infrastructure planning, navigation and transportation through waterways, river restoration, among many others. The diverse and often inspirational water features seen in rivers, from breath-taking rapids and waterfalls to lowland meanders, are mainly controlled by their underlying bathymetry (Garcia, 2008; Julian, 2018; Robert, 2003). Many measurement techniques, ranging from as simple as a rod to acoustic sounding, and more advanced technologies such as close-range photogrammetry and remote sensing, have been used to obtain bathymetric information at a point or over an area (Chen et al., 2019; Collins et al., 2020; Lane & Chandler, 2003).

However, despite the advancements in measurement techniques, there are still technological, economical, and logistical barriers in the use of bathymetry measurement and sensing to obtain high-resolution data over wide spatial and temporal ranges (Casas et al., 2006; Fonstad & Marcus, 2005; Lee et al., 2018).

Alternatively, indirect methods, such as bathymetry inversion from surface velocity, are attractive and have gained substantial interests (Almeida et al., 2018; Andreadis et al., 2007; Fonstad & Marcus, 2005; Ghorbani-dehno et al., 2021; Lee et al., 2018; Wilson & Ozkan-Haller, 2012). This work presents one such method which uses a fast deep-learning-based surrogate model, in place of often slow physics-based models, to accelerate the inversion process. More importantly, here we employ a hyper-efficient inversion procedure based on gradient descent, automatic differentiation and backpropagation. Here, “hyper-efficient” refers to the fact that the inversion process only takes seconds. The training of the surrogate model is not efficient. To enable backpropagation, we make use of the fact that the neural network surrogate is already differentiable, in a similar spirit to the recently proposed differential parameter learning (dPL) (Aboelyazeed et al., 2023; Bindas et al., 2024; Feng et al., 2022, 2023; Tsai et al., 2021). The gradient obtained in such automatic and fast fashion is then ingested by an inversion optimizer.

Bathymetry inversion has been reported in several previous researches with very encouraging results. Like most inversion problems, the solution of bathymetry inversion is ill-posed, which is characterized by the questions of existence, non-uniqueness, and stability in response to perturbations in measurement data. Previous studies have explored the use of different approaches to obtain useable solutions and deal with the ill-posed nature of...
Two-dimensional SWEs and their variants are the backbone of many models for hydrological and hydraulic predictions. These models typically use traditional methods such as the finite volume method (FVM) to discretize the governing equations on a mesh. They are often called physics-based models (PBMs), for example, SRH-2D by U.S. Bureau of Reclamation (USBR) (Lai, 2010), HEC-RAS 2D by U.S. Army Corp of Engineers (USACE) (Brummer, 1995), FLO-2D (O’Brien, 2011), RiverFlow2D (Hydronia, 2016), MIKE 21 (Warren & Bach, 1992), TUFLOW (Huxley & Syme, 2016), LISFLOOD (Van Der Knijff et al., 2010), PAWS (Shen & Phanikumar, 2010), among many others. They have been widely used in both academic researches and practice. However, for bathymetry inversion purpose, these PBMs are computationally expensive. A popular alternative is to build surrogate model which can approximate the dynamics between model input and output, at a small fraction of the computational cost of their corresponding PBMs.

Deep learning (DL), especially deep neural network (DNN), is one of the most popular ways for building surrogate models due to its capability of capturing high-dimensional nonlinearity (LeCun et al., 2015; Shen, 2018). Its popularity is also partially due to its ability to perform automatic differentiation. The gradient of an objective function defined on a neural network, with respect to the input, is readily available and has accuracy up to machine precision. By applying the chain rule repeatedly to elementary arithmetic operations and functions of code during code execution, automatic differentiation calculates the values of derivatives of arbitrary order with very high accuracy and efficiency (Baydin et al., 2018). Alternatives to obtain gradient either require numerical approximation, symbolic differentiation, or analytical derivation. For example, Almeida et al. (2018) used the variational inversion method where the gradient is calculated with both forward and adjoint solutions. Similar approaches have also been reported in Honnorat et al. (2009) and Larnier et al. (2021) who applied a variational data assimilation process in the optimization framework by adding the available observations (particle trajectories, water surface elevation, and river width). Although efficient, adjoint-based approach involves substantial mathematical derivations. It is also intrusive to PBMs source code because the adjoint problem needs to solve a new set of equations. Thus, the use of automatic differentiation to obtain gradient is appealing and has been swiftly adopted by many researchers for inversion problems (Ren et al., 2020; Xu & Darve, 2020; Zhu et al., 2021). Recently, Tsai et al. (2021) proposed a novel differentiable parameter learning framework to integrate big-data DL and differentiable PBMs for parameter estimation. Here, the differentiable PBMs refer to
models implemented directly using the machine learning platforms such as PyTorch (Paszke et al., 2019) or Tensorflow (Abadi et al., 2016), or their differentiable surrogates. Re-writing of PBMs with differentiable programming languages involves significant work. In addition, even with fully differentiable PBMs, the computing cost for inversion may be prohibitively high because the expensive PBMs need to be run in forward and backward modes for many iterations.

In this work, the USBR SRH-2D solver was selected as the PBM and its surrogate was built as an autoencoder using convolutional neural network (CNN). Specifically, a shared-encoder, separate-decoder architecture is used where the input image of bathymetry is encoded and then decoded to three outputs, namely, the two flow velocity components and the water surface elevation (WSE). Similar structure has been used in Guo et al. (2016) for incompressible flows around objects. In comparison with other techniques to build surrogate models, the autoencoder used in this work is more versatile and powerful to capture the nonlinear relationship between model input and output. The training data was generated by running sufficient number of SRH-2D simulations, whose input bathymetry data was randomly generated. The simple setup in this work is sufficient to prove the concept and show the feasibility of the proposed methodology. For practical use, the training data generation should take into considerations of the prior on real bathymetry.

This work is different from previous studies in the following aspects: (a) Our inversion is built upon a CNN-based surrogate and the differentiable parameter learning idea. Previous bathymetry inversion works, which showed very encouraging results, used different approaches such as variation inversion in Almeida et al. (2018), principal component geostatistical approach (PCGA) in Lee et al. (2018), and fully connected DNN in Ghorbanidehno et al. (2021). In fact, the approach in Ghorbanidehno et al. (2021) directly maps surface velocity to bathymetry, that is, there is no need for inversion. However, there is no guarantee that the inverted bathymetry agrees with physics. They also subdivided the domain into small segments and bathymetry inference was performed for each segment sequentially. In our work, the bathymetry inversion for the whole domain is performed in a “one-shot” fashion. The PCGA approach in Lee et al. (2018) requires the transformation of the irregular domain into a rectangular box. Although our example is also on a simple rectangle domain, the input and output images used by our surrogate model can embed a river of any shape. (b) We use the gradient generated from automatic differentiation. As will be shown, we examined in detail the inversion process from the initial guess to the final converged bathymetry, which resulted in valuable insights on two distinctive stages during the inversion process. (c) To overcome the ill-posedness of the inversion problem, we found that there are two necessary physical constraints on both bed elevation value and slope, which have to be embedded in the inversion loss function as two separate regularizations. A heuristic approach is proposed to solve the problem of regularization hyper-parameter determination. (d) We also investigated the effects of autoencoder architecture for surrogate model on bathymetry inversion.

The rest of the paper is organized as follows. The methodology, including the surrogate model architecture, inversion algorithm, and training data generation, is introduced first. Then the results on both the performance of the surrogate model and the inversion algorithm are presented with discussion. This paper is concluded with a summary of findings and future work.

2. Methodology

This section describes the deep-learning-based surrogate model and the inversion algorithm, followed by the introduction to the PBM solver and training data generation process.

2.1. Deep-Learning-Based Surrogate Model Architecture

The convolutional autoencoder is used to construct the surrogate model. It consists of two parts: encoder and decoder. To reduce the length of this paper, the details of the surrogate model can be found in Song et al. (2023) and the Supplementary Information. Song et al. (2023) built the surrogate to study the hydrodynamics around objects such as bridge piers in river channels. The input to the autoencoder is a signed distance field to the objects. In this work, the input is the bathymetry. In addition, Song et al. (2023) only used the surrogate for forward predictions. This paper will show how to use the differentiable surrogate in both forward and backward modes. The use of such autoencoder as a surrogate has also been reported in Guo et al. (2016) for steady-state, laminar Navier-Stokes equations and Forghani et al. (2021) for 2D SWEs. One notable difference between our work and
previous research is we used multiple output branches for different flow field variables (velocity components and water surface elevation) to investigate the effects of surrogate architecture on inversion.

The autoencoder surrogate model takes the bathymetry image, $z_b$, as the input, and outputs the predicted flow field corresponding to the input bathymetry. Indeed, the decoder has three outputs, that is, $u$, $v$, and WSE. Here $u$ and $v$ are velocity components.

To train the surrogate model, a proper loss function needs to be defined. In this work, the loss due to surrogate prediction mismatch, $L_{\text{prediction}}$, is constructed to minimize the mean squared error (MSE) of all components of the predicted flow field. The details of the loss definition can be found in the Supporting Information. The ground truth data on the flow field (ground truth) were generated from SRH-2D simulations. The SRH-2D solver has been extensively validated and it has high accuracy (Lai, 2010).

A stochastic gradient descent optimizer, ADAM, was used to update the weights and biases of the neural network (Kingma & Ba, 2014). To alleviate the vanishing gradient problem, the Rectified Linear Unit (ReLU) activation function was used (Sekar et al., 2019). For the final output layer of the decoder, the linear activation function is used to accommodate negative flow variable values.

2.2. Inversion Algorithm

The trained surrogate model was used for the inversion, that is, to find the bathymetry given a flow field. For convenience, the surrogate model can be written as

$$G z_b \quad d \quad e$$

where $G$ represents the forward model or physical simulator, $d \ (u, v, \text{WSE})$ is the flow field data, and $e$ is error due to numerical approximation and/or measurement inaccuracy. The inversion problem is then to find the inverse operator $G^1$, which is often very challenging if the problem is ill-posed, non-linear, and high-dimensional. A key strategy to stabilize the inversion is to impose additional constraints, generally referred to as regularization. Among many, Tikhonov regularization is commonly used, which adds the additional constraint on the inverted variable. Many previous works have used the zeroth-order Tikhonov regularization where the additional constraint is to minimize the norm of $z_b$. However, we will show that the zeroth-order Tikhonov regularization itself is not enough for bathymetry inversion. In this work, we propose to use two new physically based regularizations, one for the bed elevation value (a modified zeroth-order Tikhonov regularization indeed) and one for the bed slope (a first-order Tikhonov regularization). The total inversion loss function has the form of

$$L_{\text{total}} \quad z_b \quad L_{\text{prediction}} \quad \alpha_{\text{value}} \quad \alpha_{\text{slope}}$$

where $L_{\text{prediction}}$ is the flow prediction error defined in Section 2.1, $L_{\text{value}}$ is the loss due to the inverted $z_b$ values going beyond a prescribed range, and similarly $L_{\text{slope}}$ is the loss due to the slope of inverted bed exceeding a upper limit. For a specific river, the maximum bed slope is mostly determined by its sediment characteristics. $\alpha_{\text{value}}$ and $\alpha_{\text{slope}}$ are the regularization factors for their corresponding losses. Like other hyperparameters in the model, these two parameters are problem specific and need to be tuned. For inverse problems, these parameters can be determined with methods such as the L-curve criterion (Aster et al., 2013). Their determination and effects will be shown in the results section.

Essentially, the value loss $L_{\text{value}}$ and the slope loss $L_{\text{slope}}$ are used to discourage the inversion algorithm from searching solutions outside the prescribed bounds for bed elevation and slope. These kind of losses are commonly used in inversion problems for distributed parameters to find the most probable solution that reasonably matches the observation data. The slope loss here penalizes the total variation of bed elevation and is also referred to as the maximization of entropy. From the physics point of view, the slope of a sediment bed is bounded by the angle of repose, exceeding of which will cause sand slide (Song et al., 2020). To embed these prior information or constraints in the inversion process, the value and slope losses are calculated with a double-bounded ReLU (dbReLU) function (Ren et al., 2020), which has the form of

$$f_{\text{dbReLU}} x \quad \text{ReLU} x \quad x_c \quad a \quad \max 0, x \quad x_c \quad a$$
where \( x \) is the input to the function, \( x_c \) is the center of \( x \), and \( a \) is a constant which controls the range of \( x \) where the loss function is zero. It is clear that with this function as loss function, the loss will be zero within the range of \([x_c - a, x_c + a]\), that is, the zero loss will be bounded by the two ends of the range and hence the name. Away from the two end points of this range, the loss starts to increase linearly. With the above definition, the losses due to value and slope can be written as

\[
\begin{align*}
L_{\text{value}} &= \sum_{i=1}^{N_{\text{points}}} f_{\text{ReLU}}(z_b^i) \\
L_{\text{slope}} &= \sum_{i=1}^{N_{\text{points}}} f_{\text{ReLU}}(S_x^i) + \sum_{i=1}^{N_{\text{points}}} f_{\text{ReLU}}(S_y^i)
\end{align*}
\]

where \( N_{\text{points}} \) is the number of points in the input bathymetry array, \( S_x \) and \( S_y \) are the slopes of inverted bed in the \( x \) and \( y \) directions, respectively.

The method for bathymetry inversion proposed in this work involves two main steps. The first is to train the CNN-based surrogate model and the second is to perform the inversion. For the first step, the neural network is trained on a given set of data, which is made of the input/output pairs of the SWEs simulator. The training data generation will be described in the next section. The first step will result in a surrogate model \( G \). In the second step, for any arbitrary input \( z_b \), the surrogate model produces an output \((u, v, WSE)\) and the inversion loss defined in Equation 2 can be calculated. In the framework of neural network and due to its capability to perform automatic differentiation, the total inversion loss \( L_{\text{total}} \) is differentiable with respect to the input \( z_b \). The gradient \( \partial L_{\text{total}} / \partial z_b \) can then be used in a simple iterative scheme to invert the bathymetry, which can be written as

\[
\begin{align*}
z_b^{i+1} &= z_b^i - \alpha_{\text{inversion}} \frac{\partial L_{\text{total}}}{\partial z_b} \\
&= z_b^i - \alpha_{\text{inversion}} \frac{\partial L_{\text{total}}}{\partial z_b} \bigg|_{z_b} \cdot z_b^i
\end{align*}
\]

where \( i \) is the iteration number and \( \alpha_{\text{inversion}} \) is the inversion step size (learning rate). Note that \( z_b \) is a 2D array and the gradient in Equation 6 is taken element-wise. More advanced iterative schemes, such as the Gauss-Newton and Broyden–Fletcher–Goldfarb–Shanno (BFGS) methods, can be used in future work to further take advantage of the gradient and Hessian information. Equation 6 is essentially a gradient descend optimization to minimize the total inversion loss \( L_{\text{total}} \). It is noted that during the inversion process, the parameters of the surrogate neural network are frozen (not trainable). The only trainable parameters are the bed elevations \( z_{b,1} \). To start the iterations, an initial guess on the bathymetry, \( z_b^0 \), is needed. The initial guess can be samples drawn from a prior distribution and each will result in an inverted bathymetry. The ensemble can be used to assess the stability of the inversion and calculate a mean inverted bathymetry.

### 2.3. Data Generation and Preprocessing

#### 2.3.1. SWEs Solver

The training data for the surrogate model was generated with the 2D SWEs solver SRH-2D, which is a popular 2D hydraulics model (Lai, 2008). SRH-2D is developed by the U.S. Bureau of Reclamation for many of the nation’s large water resources planning and design projects. It is also adopted by the U.S. Federal Highway Administration for the design and protection of transportation infrastructure against flooding. The method developed in this work is generic and can be used with training data produced with other SWEs solvers.

The SWEs can be derived by depth-averaging the 3D Navier-Stokes equations. They have the following general form (Lai, 2010):

\[
\begin{align*}
\frac{\partial h}{\partial t} + \frac{\partial hu}{\partial x} + \frac{\partial hv}{\partial y} &= 0 \\
\frac{\partial hu}{\partial t} + \frac{\partial huu}{\partial x} + \frac{\partial hv}{\partial y} &= \frac{\partial hT_{sx}}{\partial x} + \frac{\partial hT_{sy}}{\partial y} + \frac{gh}{\rho} \frac{\partial z_b}{\partial x} - \frac{\tau_{bx}}{\rho} \\
\frac{\partial hv}{\partial t} + \frac{\partial huv}{\partial x} + \frac{\partial hv}{\partial y} &= \frac{\partial hT_{sx}}{\partial x} + \frac{\partial hT_{sy}}{\partial y} + \frac{gh}{\rho} \frac{\partial z_b}{\partial y} - \frac{\tau_{by}}{\rho}
\end{align*}
\]
Figure 1. Four example synthetic bathymetries generated with 2D Gaussian process. The two dashed lines on Sample 0 are the two profiles, longitudinal and cross-sectional, used later in this paper. Flow is from left to right.

\[
\frac{\partial h}{\partial t} + u \frac{\partial h}{\partial x} + v \frac{\partial h}{\partial y} = \frac{\partial}{\partial x} \left( \frac{\partial h}{\partial x} \right) + \frac{\partial}{\partial y} \left( \frac{\partial h}{\partial y} \right) - \frac{g h}{\rho} \frac{\partial^2 z}{\partial t^2} + \tau_{bx} \tau_{by}
\]

where \( u \) and \( v \) are the depth-averaged flow velocities in \( x \) and \( y \) directions, respectively; \( h \) is the water depth; \( z_a \) and \( z_b \) is the bed elevation; \( T_{sx}, T_{sy}, \) and \( T_{xy} \) are the depth-averaged turbulence stresses; \( \tau_{bx} \) and \( \tau_{by} \) are the bed shear stresses in \( x \) and \( y \) directions, respectively; \( g \) is the gravitational acceleration; \( \rho \) is the water density. The bed stresses, \( \tau_{bx}, \tau_{by} \), are estimated by the Manning’s resistance equation as:

\[
\tau_{bx,by} = \rho C_f \frac{u^2}{n} \quad v^2 \quad u, v
\]

where \( \rho \) is the density of water, \( C_f = gn^2/h^{1/3} \), and \( n \) is the Manning’s roughness coefficient.

More details about the turbulence model and the physical means of all terms can be found in Rodi (1993) and Lai (2008).

2.3.2. Training Bathmetry Generation

There are different ways to generate the bathmetry for training cases. Among them, the most popular way uses a stochastic process which can embed some prior knowledge on the bed. In this work, the training bathmetry is randomly drawn from a 2D Gaussian process. To control the properties of the generated bathmetry, such as overall bed elevation range and feature size, the parameters of the Gaussian process can be tuned. Additionally, if ground-truth bed elevations at certain locations are available, they can be assimilated into the Gaussian process prior to the draw. Following Landon et al. (2014), this work uses the Radial-Basis Function (RBF) as the 2D Gaussian kernel, which has the form of

\[
k(x,x) = \sigma^2 \exp \left( -\frac{\Delta x^2}{2\ell^2} - \frac{\Delta y^2}{2\ell'^2} \right)
\]

where \( x \) and \( x \) are two arbitrary points in the domain; \( \Delta x \) and \( \Delta y \) are their distance in \( x \) and \( y \) directions, respectively; \( \sigma^2 \) is the prior variance; \( \ell \) and \( \ell' \) are the length scales in \( x \) and \( y \) directions, respectively.

The parameters for the kernel should be set based on some basic prior information about the bathmetry. For demonstration purpose, this work generated hypothetical bathmetries as training data. The bathmetries were first generated on a unit square. The prior variance \( \sigma^2 \) has a value of 0.5. The length scales \( \ell \) and \( \ell' \) have the values of 0.1 and 0.2, respectively. Then, the bathmetries on the unit square were mapped to a rectangular domain with a length of 25.6 m and a width of 6.4 m. These parameters were set such that the generated bed elevation is in the range of 0.5–0.5 m and there is approximately one or two significant bed features (bedform-like) in the simulation domain. Three thousand bathmetries were generated in this work. Figure 1 shows four samples and
flow is from left to right. For example, Sample 0 shows one elongated bar on the top and one hump at the outlet. There is also a deep pool near the outlet hump. Sample 0 will be referenced later as an example for some detailed analysis.

### 2.3.3. Flow Data Generation, Parameter Tuning, and Implementation

Three thousand simulation cases were run with the generated bathymetries, out of which 80%, 15%, and 5% were used for training, validation and testing, respectively. All cases were run with the same conditions except the bathymetry. Steady flow in the rectangular channel was simulated. The inflow discharge was 3 m³/s and the downstream was set with a fixed water surface elevation of 1 m. The computational domain was discretized into 1,580 triangles with an average size of 0.5 m. A constant Manning’s roughness $n$ of 0.03 was used. All simulations were run until steady state. All simulation results and input bathymetries were then sampled on a grid of 32 rows and 128 columns, which is proportional to the width-to-length ratio of the channel. Data for training and inversion have the same image resolution (32 by 128). The grid resolution is double that of the computational mesh to ensure the capture of bathymetry and flow field details. It is noted that all the data were made dimensionless with the minimum and maximum values of each variable for model training and evaluation (termed min-max normalization in ML). However, in the plots and discussion of this paper, the variables are converted back to dimensional form for more straightforward understanding.

The performance of surrogate model and the inversion highly depends on the choice of hyperparameters. The optimal values for these hyperparameters were obtained through manual tuning, which is manageable because of the relative simplicity of the hypothetical cases in this work. Automatic and more intelligent hyperparameter optimization will need to be performed if the cases are more complex. The CNN-based surrogate model used in this work contains two convolution layers in the encoder and four transpose convolution layers in the decoder for each of the outputs. The fully connected layer for the code has a length of 1,024. Other parameters of the surrogate model can be found in Table 1.

During the surrogate training process, to update the weights and biases in the neural network, an initial learning rate was set at 10⁻³. A learning rate scheduler was utilized to reduce the learning rate by a factor of 0.5 when the validation loss did not decrease over 5 epochs. One hundred epochs with a mini-batch size of 10 were performed until both training and validation losses were simultaneously minimized. For inversion, a fixed learning rate of 0.01 was used to perform 1,000 steps of iterations which has been proven to be sufficient to obtain converged solutions.

The surrogate model and the inversion method were implemented in Tensorflow (Abadi et al., 2015) with Keras API (Chollet, 2015). The training of the surrogate model and the inversion were performed on Amazon AWS with a NVIDIA K80 GPU card. The simulations with the physics-based model SRH-2D were performed on a desktop with an Intel Core 3.40 GHz CPU. The 3,000 SRH-2D simulations took about 24 hr to finish. The training of the surrogate model took about 2 hr. Each inversion took about 1 min. The code and data generation scripts used in this work can be accessed at https://github.com/psu-efd/dl4HM/tree/main/examples/bathymetry_inversion_2D.

| Unit   | Layers                                                                 | Size $(H, W, N_t)$ or number of neurons |
|--------|------------------------------------------------------------------------|----------------------------------------|
| Input $z_n$ | Conv2D, 128 filters of size (8,8), stride (8,8), padding "same"     | (32,128,1)                             |
| Encoder | Conv2D, 512 filters of size (4,4), stride (4,4), padding "same"     | (4,16,128)                             |
| Encoder | Flatten                                                               | (2,048,1,1)                            |
| Fully connected | Dense                                                                 | (1,024)                                |
| Reshape | Reshape                                                               | (1,1,1,024)                             |
| Decoder | Conv2DTranspose, 512 filters of size (8,8), stride (8,8), no padding | (8,8,512)                              |
| Decoder | Conv2DTranspose, 256 filters of size (8,2), stride (8,2), no padding | (64,16,256)                            |
| Decoder | Conv2DTranspose, 1 filter of size (2,2), stride (2,2), no padding   | (128,32,1)                             |
| Decoder | Permute with order (2,1,3)                                            | (32,128,1)                             |
Because of the large number of simulations needed, and to automate the data generation and processing, the python package, Python-based Hydraulic Modeling Tools, pyHMT2D, was used to control SRH-2D modeling runs and transform the results to the inputs and outputs of the neural network (Liu, 2021). Source code of pyHMT2D can be found on GitHub (https://github.com/psu-efd/pyHMT2D).

3. Results and Discussions

3.1. Performance of Surrogate Model

The premise of accurate inversion using surrogate model is the accuracy of the surrogate model itself. In this work, the surrogate model was evaluated with three metrics, that is, the MSE of model prediction, the root-mean-square error (RMSE) of absolute error and relative error. The relative error for a grid point is defined as the ratio of the absolute error to the maximum of the predicted and ground-truth values (to avoid division by zero problem). Taking \( u \) as an example, the RMSE of absolute error \( e_{m,u} \) and relative error \( e_{r,u} \) for each case are defined as follows:

\[
\begin{align*}
E_{m,u} &= \frac{1}{N_{\text{points}}} \sum_{i=1}^{N_{\text{points}}} u_i - \bar{u} \\
E_{r,u} &= \frac{1}{N_{\text{points}}} \sum_{i=1}^{N_{\text{points}}} \frac{|u_i|}{\max(|u_i|, |\bar{u}|)}
\end{align*}
\]

where \( N_{\text{points}} \) is the number of points in each test case. Then the mean, max, and standard deviation of the two errors for all test cases are calculated.

Figure 2 shows the history of the training and validation MSE losses over 100 epochs. Both losses were driven to their minimums and there was no significant over-fitting because the validation loss is close to the training loss. This shows the architecture of the CNN surrogate and its hyperparameters are proper for the given dataset.

The surrogate model proposed in this work is of high accuracy and properly captures the input-output dynamics embedded in the SWEs solver. Figure 3 shows an example case comparing the results from SRH-2D and the surrogate. The bathymetry of this case is the “Sample 0” shown in Figure 1. The result shows that the difference between the results of PBM and surrogate is small. For this particular case, the RMSEs of absolute error for \( u \), \( v \) and WSE are 0.0076 m/s, 0.0066 m/s, and 0.0007 m, respectively.

Figure 2. Training and validation losses.

Figure 3. An example case showing the performance of CNN surrogate model. The bathymetry for this case is the “Sample 0” shown in Figure 1. The first and second columns show the contours of velocity components \( u \), \( v \), and WSE from SRH-2D and CNN surrogate, respectively. The third column shows the differences.
The prediction error for WSE is smaller, in both absolute and relative senses, than the errors for the two velocity components. This is not surprising because the WSE field is typically much smoother and has less variations than the velocity field. In fact, because of the hydrostatic assumption in SWEs, the water depth \( h \), which directly controls WSE, is equivalent to the pressure in the Navier-Stokes equations. In the Navier-Stokes equations, the pressure is governed by an elliptic Poisson’s equation, which has self-smoothing effect over time and space for proper initial and boundary conditions.

The relatively small prediction error in WSE may have some implications for inversion. For a given bathymetry, it is easy for the surrogate model to make more accurate WSE predictions than the velocity. In other words, the total error is dominated by the velocity errors. If this is true, then during inversion the WSE error does not contribute as much as the velocity error. Therefore, it is possible to drop WSE and only use velocity components \( u \) and \( v \) for the inversion. This is good news because that means we need less information to invert the bathymetry. In practice, it is difficult to obtain velocity and WSE at the same time. Although the CNN surrogate model has three output branches (\( u, v \) and WSE), we do not need to use all of them to do the inversion. More discussion will be presented in the inversion result section.

The spatial distributions of prediction errors shown in Figure 3 do not show any clear trend in where large errors are located. For example, locations of large prediction errors in the velocity components do not seem to be coincident with either low or high of bathymetry and velocity themselves. These observations apply to all test cases. It might be beneficial in future research to investigate what drives the spatial distribution of prediction error from the CNN and whether it has implications for inversion.

The statistics of the flow prediction errors for all 150 test cases are shown in Table 2. The statistics include the mean, max, and standard deviation (std). The prediction errors for all test cases are small, except the relative error for the velocity component \( v \), which has a mean value of about 39.6% and a maximum of about 61.6%. This high relative error should not cause any alarm because it is due to the small magnitude of \( v \) in the flow results. The absolute error for \( v \) is reassuringly very small.

The accuracy of the surrogate model was further analyzed with the distribution of the L2 norm of the prediction error \( e \) in Equation 1. The histogram, the fifth, 50th, and 95th percentiles of the L2 norms for all 150 test cases are shown in Figure 4. The L2 norm of surrogate prediction error \( e \) is used in the determination of the inversion loss parameters \( \alpha_{\text{value}} \) and \( \alpha_{\text{slope}} \). Here, the L2 norm only has the contributions from \( u \) and \( v \) prediction errors because only velocity is typically used for inversion. From the figure, it is clear that the surrogate model error is in a range (approximately from \( 3 \times 10^{-1} \) to 1.0) and each case’s error is different. Its implication for inversion is discussed next.

### 3.2. Performance of Bathymetry Inversion

#### 3.2.1. Inversion Parameter Determination

The performance of inversion also heavily depends on the parameters. The values of relevant parameters for inversion are listed in Table 3. Indeed, some of the parameters have physical meanings and should reflect the specific problem setup. For example, the mean and amplitude of bed elevation value and slope should be set with our prior knowledge of the bathymetry. In this case, the normalized bed elevation should be in the ranges of \([0.5, 0.5]\). The bed slopes in \( x \) and \( y \) directions should be in \([0.08, 0.08]\) and \([0.15, 0.15]\), respectively. In practice, this prior information has to be obtained from other means such as survey or historical maps.

Among all the parameters, \( \alpha_{\text{value}} \) and \( \alpha_{\text{slope}} \) defined in Equation 2 are the most difficult to be determined. Their values effectively balance the surrogate prediction error and regularizations in the total inversion loss \( L_{\text{total}} \). Among other things, they depend on the surrogate model error \( e \). For inverse problems, the “L-curve” criterion is often used to determine their values (Aster et al., 2013). When plotted on a log-log scale, the curve of model prediction

#### Table 2

| Flow variable | RMSE of absolute error (m/s for \( u \) and \( v \) and \( m \) for WSE) | Relative error (%) |
|---------------|---------------------------------------------------------------|--------------------|
| \( u \)       | Mean: 0.0079, Max: 0.0169, Std: 2.27e-03                     | Mean: 1.3155, Max: 2.7425, Std: 0.3786 |
| \( v \)       | Mean: 0.0066, Max: 0.0246, Std: 2.06e-03                     | Mean: 39.6035, Max: 61.5791, Std: 4.7272 |
| WSE           | Mean: 0.0007, Max: 0.0035, Std: 2.98e-04                     | Mean: 0.0554, Max: 0.2060, Std: 0.0204 |

Figure 4. Histogram of the L2 norms of surrogate prediction errors for all 150 test cases. Here the L2 norm only has the contributions from \( u \) and \( v \) prediction errors.
loss versus regularization loss for a linear problem often has a “L” shape, hence the name. However, for nonlinear problems, the curve often shows more complicated shapes. In addition, the “L-curve” criterion only works for one parameter. In this work, we found that the parameter of bed elevation value, $\alpha_{\text{value}}$, is not sensitive. Its function is to tamp down bed elevations exceeding the specified range. In addition, the smoothing effect of the slope regularization also has certain effect to reduce the extreme bed elevations. Through trial and error, a value of 0.1 for $\alpha_{\text{value}}$ was found to be appropriate. Therefore, we will focus on $\alpha_{\text{slope}}$ next.

The value of $\alpha_{\text{slope}}$ is determined through a heuristic approach inspired by the “L-curve” criterion for linear problems. For each case, we performed the inversion using 10 different values of $\alpha_{\text{slope}}$ in the range of [0.01, 31.6]. For each $\alpha_{\text{slope}}$ value, 11 starting beds $\mathbf{x}_0$ were randomly generated. This number of 11 was determined through trial and error, which served the purpose of this research. More initial guesses can be used in reality and the mean inverted bed will be smoother. So for each case, we performed 110 inversions. Their corresponding flow prediction losses $L_{\text{prediction}}$ and slope regularization losses $L_{\text{slope}}$ are plotted in Figure 5. To reduce clutter, only a subset of the data is plotted. The mean of all 11 inversions for each $\alpha_{\text{slope}}$ value is plotted as a larger marker with edge. All the mean points are connected to show the “L-curve”. The figure shows that the curve does not have an

| Hyperparameter | Value |
|----------------|-------|
| Learning rate $\alpha_{\text{inversion}}$ | 0.01 |
| Value loss regularization factor $\alpha_{\text{value}}$ | case dependent |
| Bed elevation mean for value loss regularization $x_{\text{value}}$ | 0.0 |
| Bed elevation amplitude for value loss regularization $a_{\text{value}}$ | 0.5 |
| Slope loss regularization factor $\alpha_{\text{slope}}$ | case dependent |
| Slope mean for slope loss regularization in both x and y directions $x_{\text{slope}}$ | 0.0 |
| Slope amplitude for slope loss regularization in x direction $a_{\text{slope}}$ | 0.08 |
| Slope amplitude for slope loss regularization in y direction $a_{\text{slope}}$ | 0.15 |

Figure 5. Determination of slope loss parameter $\alpha_{\text{slope}}$ for an example case. This case corresponds to “Sample 0” in Figure 1. The scatter markers are colored by their corresponding $\alpha_{\text{slope}}$ values, which are annotated in the figure. For each $\alpha_{\text{slope}}$ value, 11 inversions were performed with different initial guesses of the bed. The mean of all 11 inversions for each $\alpha_{\text{slope}}$ value are plotted as a larger marker with edge. All the mean points are connected to show the “L-curve”. For the 11 cases of each $\alpha_{\text{slope}}$ value, the confidence ellipse with two standard deviations is also drawn to show the clustering. The percentiles of the surrogate prediction errors are also shown.
“L” shape due to nonlinearity. For the 11 cases of each $\alpha_{\text{slope}}$ value, the confidence ellipse with a radius of two standard deviations is also drawn to show the clustering.

Unlike in linear cases where there is a clear corner point on the “L-curve” for both $L_{\text{prediction}}$ and $L_{\text{slope}}$ to reach their respective minimums, there is no such point for the nonlinear case shown in Figure 5. Here, we propose a heuristic approach which is based on the following concurrent requirements:

- minimize prediction loss $L_{\text{prediction}}$. The figure shows that the prediction loss decreases as $\alpha_{\text{slope}}$ decreases. This is expected because in the limit of $\alpha_{\text{slope}}$ 0, the surrogate model makes the most accurate prediction within the bound of its accuracy. Based on this requirement, we may choose a small $\alpha_{\text{slope}}$ value such as 0.01.
- minimize the scatter (uncertainty) in prediction loss $L_{\text{prediction}}$. The figure shows that the scatter in prediction loss $L_{\text{prediction}}$, measured by the horizontal size of the confidence ellipse, decreases as $\alpha_{\text{slope}}$ decreases. This is again as expected because the surrogate model performs best with no extra terms added to the loss function. Based on this requirement, we may again choose a small $\alpha_{\text{slope}}$ value such as 0.01.
- minimize the scatter (uncertainty) in slope loss $L_{\text{slope}}$. The figure shows that the scatter in slope loss $L_{\text{slope}}$, measured by the vertical size of the confidence ellipse, is the largest when $\alpha_{\text{slope}}$ has a value of 0.01. For other values, the scatter is comparable.

Balancing all the above requirements, the $\alpha_{\text{slope}}$ value of 0.1 seems most reasonable. In addition, the prediction losses $L_{\text{prediction}}$ corresponding to this $\alpha_{\text{slope}}$ value are within the 5th and 95th percentiles of the surrogate model prediction accuracy. Results beyond the range bounded by the 5th and 95th percentiles do not make sense. They either overfit or underfit the $\alpha_{\text{slope}}$ value to drive the surrogate model out of its accuracy range.

### 3.2.2. Inversion Result Evaluation

This section shows the bathymetry inversion results. Like in many previous studies, the inversion discussed in this section only used the velocity $u$ and $v$ as the input (Almeida et al., 2018; Landon et al., 2014; Lee et al., 2018). From the practical point of view, this is reasonable because velocity data can be obtained relatively easily. The effects of inclusion or exclusion of WSE in the inversion is discussed later.

As an example, Figure 6 shows the inversion results for one of the cases. All these inversion cases are in the test dataset, not in the training and validation datasets. Note here the truth bathymetry is again the “Sample 0” shown in Figure 1. The top row of Figure 6, from left to right, shows the $z_0$ truth, mean of all inverted $z_0$ fields from all 11 initial bathymetries, and the differences between truth and mean inverted beds. The rest rows show three individual examples of inversion results started from different initial beds. Note that for the second row the initial bed is flat with random white noise.

**Figure 6.** Example inversion results for $z_0$ (the case corresponds to “Sample 0” shown in Figure 1). The top row, from left to right, shows the $z_0$ truth, mean of all inverted $z_0$ fields from all initial conditions, and the differences between truth and inverted beds. The rest rows show three individual examples of inversion results started from different initial beds. Note that for the second row the initial bed is flat with random white noise.
three individual examples of the inversion from three different initial bathymetries, including the one from the initial flat bed with random white noise. The results show that all inversions for this particular case converged to similar, though subtly different, final bathymetries.

The inverted bathymetries not only qualitatively, but also quantitatively, recover the truth. For the example shown in Figure 6, all inversions in the 11 ensemble capture the elongated bar on the top and the hump at the outlet. In addition, they also capture the deep pool feature near the outlet. Quantitatively, the RMSE of the inversion error is about 0.10 m for the mean and about 0.12 m for individual inversions. The slight improvement by the mean shows the benefits of using an ensemble, instead of a single inversion, because of the uncertainties.

To further appreciate the efficacy and uncertainty of the inversion method proposed in this work, Figure 7 shows the profiles of the inverted beds for the same case shown in Figure 6. The locations of the two profiles, one longitudinal and one cross-sectional, are shown as the black dashed lines on bathymetry “Sample 0” in Figure 1. The two profiles highlighted with heavy weight lines are for the truth and the mean of all inversions. The ensemble of inverted bed profiles is blended in the background with light weight lines to show the band of uncertainties. The mean of inverted bed profiles generally follows the profiles of truth. The uncertainty band approximately bounds the truth for both longitudinal and cross-sectional profiles, which is an evidence for the reliability of the inversion result.

### 3.2.3. Inversion Process Analysis

Some detailed analysis was performed on the inversion process to shed light on how the iterative inversion algorithm finds the solution. Figure 8 shows the inverted bathymetry at eight iteration steps for the same example case shown in Figure 6. This inversion example started using the initial flat bed with random noises. An animation
of this inversion example is in the Supplementary Information. To aid the analysis, the total inversion loss and its components are plotted in Figure 9.

The inversion process clearly shows two stages. The first stage is between Iteration 0 to about 100. This initial stage is characterized by the rapid change of bed with blocks or bands such that it can quickly evolve to a state that can describe the overall landscape of the target bathymetry. This can be easily observed in the left column of Figure 8 where the inverted beds at Iteration 0, 10, 50, and 100 are shown. The initial inverted beds show some hint on the two bed bars at the top and the outlet, as well as the deep pool at the outlet (the truth bathymetry is “Sample 0” in Figure 1). The first stage can also be easily identified in the loss history plotted in Figure 9. During this stage, the total loss and its three components all drops quickly. It is noted that at Iteration 0, the regularization losses due to bed elevation value and slope are zero. However, after the first iteration which has not taken into account the regularizations yet, the inverted bed has significant violations of the imposed bounds for bed elevation and slope. Thus, the losses for value and slope regularizations have a sudden jump at Iteration 1. The inverted beds from the first several iterations are far from the truth. Thus, the loss due to errors in the predicted flow field is also high. The continuation of the inversion iterations drives down all the losses efficiently because the total loss is very responsive to the change of bed, that is, the gradient $\partial L_{\text{total}} / \partial z_0$ has very large magnitude in the first stage.

The blocks or bands of the inverted beds at the beginning of the inversion process is due to the convolutional nature of filters in the encoder. Specifically, the length scale of the blockiness or band is directly proportional to the filter and stride sizes in the convolutional layer, especially the first one. Some portion of information on the bathymetry, in particular the spatial scales smaller than the scale defined by the filter and stride sizes, is lost during encoding process. This is one inherent limitation of using image-based regressions such as CNN. To demonstrate, Figure 10 shows one example bed, two feature maps in the first convolution layer of the encoder corresponding to this bed, and the inverted bed at one early iteration. The shown feature
maps are two representatives out of the 128 feature maps in the first convolution layer. One can observe that the two feature maps roughly capture some bed peaks and depressions. It is also clear that the inverted bed at the shown iteration has similar pattern of feature map 0. Intuitively, the inversion is initially guided by the large-scale, main features of the bed embedded in the feature maps.

One could argue that to reduce the blockiness or bands of the initial inverted beds, we can reduce the sizes of filter and stride, and add more feature maps and layers in the CNN surrogate model. However, this naive approach will also increase the complexity of the CNN architecture and the trainable parameters will increase exponentially. For a given training dataset, the surrogate model with increased complexity will quickly overfit and make the surrogate model prediction useless. Therefore, there is a trade-off between the accuracy of forward prediction and inversion using CNN-based surrogate models. It is beyond the scope of this work to investigate whether we can and if so, how to overcome this dilemma. One potential solution is to use surrogate models for point-to-point, instead of image-to-image, predictions. In other words, we can build surrogate models without the use of CNN technique. Recently, such surrogate model has been proposed in Song et al. (2023).

Fortunately, the blockiness or band of inverted beds at the beginning of inversion is not a big concern. The banded bed will be smoothed out in the second stage which happens roughly after Iteration 100. As shown in the right column in Figure 8, the blocks and edges of the inverted bed are gradually smoothed out. In this process, more small-scale bathymetric details emerge to better depict the final inverted bed. All these are driven by the continued minimization of the total loss, which needs to minimize all three loss components. From Figure 9, it is important to note that in this second stage (after Iteration 100), the prediction loss \( l_{\text{prediction}} \) stays almost constant. The value regularization loss oscillates with negligibly small values, which indicates that inverted bed elevation is well within the imposed bound. What makes significant changes is the slope regularization loss. Indeed, the bed slope changes mostly happen at the edge of the large-scale blocks resulted from the first stage. The slope regularization resembles the detailing and finishing of a rough sculpture.

### 3.3. Effects of Inversion Regularization

Regularization is the key to achieving useable inversion results. This section will show the effects of the two regularizations proposed in Section 2.2. Four cases will be compared, that is, with both value and slope regularizations, with slope regularization only, with value regularization only, and with no regularization.

Figure 11 shows the contours of inverted bathymetries for the four cases. The RMSE of inverted \( z_b \) is also reported on the figure. Figure 12 plots the longitudinal and cross-sectional profiles of the inverted bathymetries. Again, the truth bathymetry is the “Sample 0” in Figure 1. The results show that without proper regularization, the inversion produces suboptimal or even garbage solutions in comparison with the truth. With slope regularization, the inverted bathymetry is close to the truth. However, because of no constraint on value, the inverted bed elevation is beyond the specified range in many places (see the portion of bright white and deep blue pixels beyond the range of colorbar in Figure 11b). With only value regularization, the inverted bathymetry shows more
blockiness and noisiness. The noisiness can also be observed in the profiles shown in Figure 12. The worst result among the four is the one with no regularization, whose RSME is the highest.

Although not plotted, the results also show the prediction losses $L_{\text{prediction}}$ for both cases of slope regularization only and value regularization only are comparable. That means the inverted bathymetries shown in Figures 11b and 11c are both admissible solutions if the flow prediction loss is the only metric. This is a clear evidence of the non-uniqueness for the bathymetry inversion problem as defined in this work. To shrink the solution space and nudge the inversion toward a useable solution, physical constraints in the form of inversion loss regularizations must be utilized. The result show that the constraint on the bed elevation value itself using the zeroth-order Tikhonov regularization is not sufficient. Additional constraint on the bed slope, that is, the smoothness of the inverted bathymetry, is also critical. For practical applications, these additional constraints on value and slope should be adjusted based on prior knowledge or belief of the bathymetry to be inverted.

### 3.4. Inversion Uncertainty

Because of the non-uniqueness and ill-posed nature of bathymetry inversion, uncertainty quantification is of great interest for practical purpose. To demonstrate the use of the inversion method proposed in this work, a simple uncertainty study was performed. The inversion input, that is, velocity, was augmented with 10% uncertainty by adding random perturbations. Two hundred inversions were performed with the randomly perturbed velocity fields and then statistics were calculated for the inverted bathymetries.

Figure 13 shows the mean inverted longitudinal and cross-sectional profiles with uncertain velocity fields. The 95% confidence intervals around the mean profiles are shown as shaded areas. The truth bed profiles are shown as black solid lines. The case uses the “Sample 0” bathymetry in Figure 1 as the truth. The result shows that the truth bed profiles are bounded by the 95% confidence intervals, indicating the efficacy of the proposed inversion method and the appropriateness of the hyperparameter values.

### 3.5. Effects of CNN Surrogate Architecture

The architecture of CNN surrogate model affects the forward prediction. It is also important to know how the architecture affects the inversion. This section discusses two aspects of this effect.

One aspect is that so far we only used two outputs of the surrogate model, that is, two velocity components $u$ and $v$, in the inversion. The question is then whether it has any advantage to design a CNN surrogate which only has two outputs for $u$ and $v$. To distinguish, we denote the surrogate with three outputs as $\text{NN}_{(u,v,WSE)}$ and the one with only two outputs as $\text{NN}_{(u,v)}$. Theoretically, the training of the two surrogate models with the same dataset will result in different parameters (weights and biases) in the neural nets. Consequently, the gradient of $\partial L_{\text{total}}(\mathbf{z}_b)/\partial \mathbf{z}_b$ in Equation 6 will be different. The surrogate model $\text{NN}_{(u,v)}$ was trained using the same dataset except that it only has two outputs for $u$ and $v$, not WSE.
The second aspect is whether the inclusion of WSE in the inversion, in conjunction with \( u \) and \( v \), makes any difference. In Section 3.1, we already discussed the relatively small prediction error for WSE in comparison with those for \( u \) and \( v \). We hypothesize that the inclusion of WSE in inversion does not have significant contribution for the inversion, at least for the problem defined by the dataset in this work.

Based on above discussion, we compare three different approaches, namely inversions using \((u, v)\) from \(\text{NN}_{(u,v)}\), \(WSE\), \((u, v, WSE)\) from \(\text{NN}_{(u,v,WSE)}\), and \((u, v)\) from \(\text{NN}_{(u,v)}\). Figure 14 shows the inverted bathymetries using the three approaches. Each subplot shows the mean of the bathymetries inverted from 11 different starting guesses of the bed. Although slightly different, all three approaches produce similar bathymetries with comparable RMSE.
Figure 13. Profiles of the inverted bed with 10% uncertainty added to the velocity field: (a) Longitudinal profile in the middle of the channel, (b) Cross-sectional profile at half channel length. The 95% confidence intervals around the mean are shown as the shaded areas.

For the surrogate model NN\textsubscript{(w,v,WSE)}, the inversions with and without WSE produce almost identical results, which confirms our hypothesis regarding the importance of WSE for this problem. Whether this conclusion can be generalized is unclear at this point. It may so happen that \( u \) and \( v \) are sufficient to do the inversion and WSE is redundant for this particular problem. For a different problem where WSE can contribute more new information, its inclusion may be necessary.

Judging by the comparable RMSE of inverted bathymetry, the inversion using \((u, v)\) from NN\textsubscript{(w,v)} has no advantage than that using \((u, v)\) from NN\textsubscript{(w,v,WSE)}. Future work needs to investigate the generalizability of this conclusion. Again, for cases where WSE can contribute unique information, the inclusion of WSE might be necessary and therefore inversion using \((u, v)\) from NN\textsubscript{(w,v)} might perform poorly.

4. Real-World Case Demonstration

The simple case of a rectangular channel shown in the previous section proves the concept. In reality, river channels often meander and have irregular shapes. To further demonstrate the feasibility of the methodology proposed in this paper, a real-world application case is shown. Like many methods in the ML/AI world, some good “engineering” is needed for them to transition from simple toy problems to complex real problems. In the context of this work, the main issue is that CNN needs the data to be rectangular images. However, rivers usually have irregular shapes. Direct sampling of simulation data in physical space on a rectangular grid is not the most effective way. In fact, we tried this approach and the result was not great. The main reason is that direct sampling in physical space may waste a large portion of the images (not occupied by modeling domain). In other words, the information contained in CNN training images is reduced. To deal with that, a transformation is developed to map the data between the physical \( x-y \) space to a space spanned by the streamwise and cross-sectional coordinates \( s-t \). Many aspects of the workflow, such as training data generation and loss definitions, have to be modified. Due the
Figure 14. Effects on inversion using different CNN surrogate architectures and different inversion input: (a) inversion using $(u, v)$ from $NN_{(u,v,WSE)}$, (b) inversion using $(u, v, WSE)$ from $NN_{(u,v,WSE)}$, and (c) inversion using $(u, v)$ from $NN_{(u,v)}$.

length limit of this paper, only the major techniques and results are presented here. More details are in the Supplementary Information.

The real-world case is for a part of the Savannah River near Augusta, GA. The data was reported in Lee et al. (2018). The river section is about 1.2 km long and 100 m wide, which is located about 7.6 km downstream of the Savannah Bluff Lock and Dam. It has one 90° bend. The U.S. Army Corps of Engineers conducted a high-resolution bathymetry survey and it is shown in Figure 15a. Because of the bend, the river only occupies a fraction of the whole domain in $x$-$y$ space. The white space in Figure 15a is not used. If we sample the CNN training images in $x$-$y$ space, the information density contained in these images will be low. To overcome that, the domain is mapped from the $x$-$y$ space to $s$-$t$ space and the resulted domain shape and bathymetry are shown in Figure 15b. It is clear that the transformed domain now occupies the major part of the rectangular bounding box. In our implementation, the Python package Shapely was used to find the $(s, t)$ coordinates corresponding to a given $(x, y)$. The line shown Figure 15a was manually drawn to generally follow the river course and it does have any particular hard constrains. The contours of the $s$ and $t$ coordinates in the river domain are shown in Figures 15c and 15d, respectively.

One of the most important steps in the workflow is to generate training bathymetry data. We used the 2D Gaussian Process in the previous simple rectangular channel case to generate synthetic bathymetry. For the real world case, we were not successful in using the same technique because it is extremely hard to specify or fit the parameters in the Gaussian process model. Instead, like in many ML/AI applications, image augmentation techniques were adopted. Specifically, image transformation in the $s$-$t$ space was performed from the base bathymetry. The main transformation was translation and it resulted in 2,080 bathymetries, some of which are shown in Figure 16.

Corresponding to the 2,080 bathymetries, SRH-2D simulations were performed to produce the flow data, that is, $u$, $v$, and $WSE$. The simulation boundary conditions were the same as those in Lee et al. (2018), that is, the inflow...
Discharge from upstream is 187.4 m³/s (left) and the downstream water surface elevation is 29.3 m (right). The simulated data were then transformed into the s-t space and sampled on a 64 by 256 grid to create training images for the surrogate. The details on the design of the surrogate, data split for training/validation/test, and hyper-parameter determination are described in the Supplementary Information. The trained surrogate model has a reasonable predictive skill. Figure 17 shows one example from the test cases. It shows the flow results from SRH-2D simulation, the CNN surrogate, and their differences. Overall, the differences are relatively small, though they seem to be slightly larger than those for the simple rectangular channel case. In addition, the surrogate model’s predicted flow field is smoother than the simulated one due to the filtering effect of CNN. In future research, the effect of filtering for real-world complex cases should be further investigated.

**Figure 15.** The Savannah River, GA case: (a) measured bathymetry in x-y space. The streamwise and cross-sectional coordinates s and 5 are also defined. (b) Measured bathymetry mapped into s-t space, (c) contour of s coordinate, and (d) contour of t coordinate.

**Figure 16.** Example bathymetries generated with image augmentation. Note that the image augmentation was performed in the s-t space and the plotted example bathymetries have been mapped back to the x-y space.
With the trained differentiable surrogate model, the bathymetry inversion step was performed with an ensemble of initial bathymetry, which were generated with random noise. The mean of the inverted bathymetries was also calculated. Figure 18 shows the inversion results. In general, the inversion produced bathymetries close to the truth, for example, the deep pool near the outer bank of the bend and the thalweg. The mean of the inverted bathymetries is much smoother than any of the individual inversion results and visually better matches with the truth.

5. Conclusion

Using a CNN-based surrogate model for a shallow water equations solver, a bathymetry inversion method is developed based on the gradient conveniently calculated with neural network’s automatic differentiation. The surrogate model uses a shared-encoder and separate-decoder architecture, which can successfully capture the dynamics between input (bathymetry) and output (flow field). To do the inversion, new regularizations have to be used for both the bed elevation value and bed slope. The new regularizations embed the prior knowledge or belief on the bathymetry to be inverted. Without these regularizations, especially the slope regularization, the inverted bathymetries shows substantial blockiness and noisiness. To enable the applications in real-world where the domain has irregular shapes, we propose to build the surrogate model in the transformed s-t space, which can better use CNN’s encoding power and improve the model prediction and inversion performance.

One of the difficulties in nonlinear inversion problem is the lack of clearly defined approach for determining the parameters of regularization losses. In this work, we found the $\alpha_{slope}$ parameter for slope loss is more important than $\alpha_{value}$ for bed elevation value. Using the “L-curve” criterion, a heuristic approach was
proposed with some guiding requirements. This approach produces optimal $\alpha_{\text{slope}}$ values and good inversion results.

The inversion process has two distinctive stages. The first is characterized by rapid changes of bed in blocks and bands such that it can quickly evolve to a state that can represent the overall landscape of the target bathymetry. The inversion loss due to flow prediction error quickly converges to its minimum during this stage. The blockiness of the inverted intermediate beds correspond to the sizes of filters and strides for the feature maps of convolution layers in encoder. During the second stage, the regularizations for bed elevation value and slope gradually smooth out the blocks and bands of bed and add more details to the inverted bathymetry. In this second stage, the inversion loss due to flow prediction error stays almost constant. It is the local adjustment of bed slope and elevation that drive the bathymetry toward its final solution. In other words, it is the two regularizations selecting the most probable solution, out of many, to fit the imposed physical constrains on the bathymetry.

The investigation on the CNN surrogate architecture reveals that different options on the CNN surrogate ($\text{NN}_{(u,v,WSE)}$ versus $\text{NN}_{(u,v)}$) and whether to include $WSE$ in the inversion yield comparable bathymetries. The fact that the use of $\text{NN}_{(u,v,WSE)}$ and $\text{NN}_{(u,v)}$ produces similar results may be due to shared-encoder and separate decoder architecture of the surrogate. In this way, the separate decoders for $u$, $v$, and $WSE$ have minimum interference. For $\text{NN}_{(u,v,WSE)}$, the inclusion of $WSE$, in conjunction with $u$ and $v$, is redundant, at least for the problem solved in this work. The root cause of this is that the prediction error for $WSE$ is much smaller than those for velocity, and thus it contributes less in the total inversion loss. This conclusion
may not be true in other cases where the prediction error for WSE responds strongly to the variation of bathymetry.

The use of surrogate model greatly reduces the computational cost of forward modeling runs during the inversion. However, this saving is at the cost of offline surrogate training time. In future work, this cost may be reduced using other approaches. For example, instead of surrogate models, an alternative option is to implement the physics-based forward models using machine learning platform and languages such as PyTorch and Tensorflow. In this way, all the operations on the inputs are recorded and automatic differentiation can be carried out directly using backpropagation. However, this alternative currently faces at least two challenges. One is the upfront cost of implementing forward models using a new language. The second is the complexity and computational cost of automatic differentiation even if we can implement physics-based models using these platforms. Solvers of SWEs using traditional methods such as the finite volume method involve tremendous amount of calculations which have to be recorded internally for automatic differentiation purpose.

Another drawback of the current method is that for a different case, all the data generation, training, and inversion steps have to be done from scratch. To address this, we may benefit from transfer learning if a new case resembles, but is not identical to, an existing case. Transfer learning starts the training process from an existing case and may substantially save the time to convergence. The same methodology can be used when the river experiences fast and significant erosion and sedimentation. Under this condition, the bathymetry will be dramatically different over time. Indeed, hydraulically speaking, the river may not be the same. The surrogate model needs be trained with a larger dataset which embeds the future river conditions or be re-trained. For real-world applications, due to the irregular shape of river corridors, more research is needed on how to generate representative training bathymetries.

Data Availability Statement

A permanent copy of code, data, and scripts used for this work has also been achieved in the CUAHSI's HydroShare: http://www.hydroshare.org/resource/357c3e413622460a91d29cc61d0ba084 (Liu, 2023). Figures were made with Matplotlib version 3.4.3 (Hunter, 2007), available under the Matplotlib license at https://matplotlib.org/.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., et al. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. (Software available from tensorflow.org). Retrieved from https://www.tensorflow.org/
Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., et al. (2016). TensorFlow: A system for large-scale machine learning. In 12th Usenix Symposium on Operating Systems Design and Implementation (osdi 16) (pp. 265–283).
Aboelyazeed, D., Xu, C., Hoffman, F. M., Liu, J., Jones, A. W., Rackauckas, C., et al. (2023). A differentiable, physics-informed ecosystem modeling and learning framework for large-scale inverse problems: Demonstration with photosynthesis simulations. Biogeosciences, 2023.
Almeida, T. G., Walker, D. T., & Warnock, A. M. (2018). Estimating river bathymetry from surface velocity observations using variational inverse modeling. Journal of Atmospheric and Oceanic Technology, 35(1), 21–34. https://doi.org/10.1175/JTECH‐D‐17‐0075.1
Andreadis, K. M., Clark, E. A., Lettenmaier, D. P., & Alsorf, D. E. (2007). Prospects for river discharge and depth estimation through assimilation of swath-altimetry into a raster-based hydrodynamics model. Geophysical Research Letters, 34(10). https://doi.org/10.1029/2007GL029721
Aster, R. C., Borchers, B., & Thurber, C. H. (2013). Parameter estimation and inverse problems (2nd ed.). Elsevier Inc
Baydin, A. G., Pearlmutter, B. A., Radul, A. A., & Siskind, J. M. (2018). Automatic differentiation in machine learning: A survey. Journal of Machine Learning Research, 18, 1–43.
Binda, T., Tsai, W.-P., Liu, J., Rahmani, F., Feng, D., Bian, Y., et al. (2024). Improving river routing using a differentiable muskingum-cunge model and physics-informed machine learning. Water Resources Research, 60(1), e2023WR035337. https://doi.org/10.1029/2023WR035337
Biondi, F., Addabbo, P., Clemente, C., & Orlando, D. (2020). Measurements of surface river Doppler velocities with along-track insar using a single antenna. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 13, 987–997. https://doi.org/10.1109/JSTARS.2020.2976529
Bradley, A. A., Kruger, A., Meselhe, E. A., & Muste, M. V. (2002). Flow measurement in streams using video imagery. Water Resources Research, 38(12), 51–51.8. https://doi.org/10.1029/2002WR001317
Brunner, G. W. (1995). HEC-RAS river analysis system. In Hydraulic Reference Manual. Version 1.0. (Tech. Rep.). Hydrologic Engineering Center Davis CA
Casas, A., Benito, G., Thorndycraft, V. R., & Rico, M. (2006). The topographic data source of digital terrain models as a key element in the accuracy of hydraulic flood modelling. Earth Surface Processes and Landforms, 31(4), 444–456. https://doi.org/10.1002/esp.1278
Chen, Y., DiBiase, R. A., McCarroll, N., & Liu, X. (2019). Quantifying flow resistance in mountain streams using computational fluid dynamics modeling over structure-from-motion photogrammetry derived microtopography. Earth Surface Processes and Landforms, 44(10), 1973–1987. https://doi.org/10.1002/esp.4624
Chollet, F. (2015). Keras. GitHub. Retrieved from https://github.com/keras-team/keras

Acknowledgments
This work is supported by a seed grant from the Institute of Computational and Data Sciences at the Pennsylvania State University. Shen would like to acknowledge support from National Science Foundation Award EAR#1832294.
Shen, C. (2018). A transdisciplinary review of deep learning research and its relevance for water resources scientists. *Water Resources Research*, 54(11), 8558–8593. https://doi.org/10.1029/2018wr022643

Shen, C., & Phanikumar, M. S. (2010). A process-based, distributed hydrologic model based on a large-scale method for surface–subsurface coupling. *Advances in Water Resources*, 33(12), 1524–1541. https://doi.org/10.1016/j.advwatres.2010.09.002

Song, Y., Shen, C., & Liu, X. (2023). A surrogate model for shallow water equations solvers with deep learning. *Journal of Hydraulic Engineering*, 149(11), 04023045. https://doi.org/10.1061/jhend8.hyeng‐13190

Song, Y., Xu, Y., & Liu, X. (2020). Physically based sand slide method in scour models based on slope-limited diffusion. *Journal of Hydraulic Engineering*, 146(11), 04020074. https://doi.org/10.1061/ASCE/HY.1943-7900.0001814

Steissberg, T. E., Hook, S. J., & Schladow, S. G. (2005). Measuring surface currents in lakes with high spatial resolution thermal infrared imagery. *Geophysical Research Letters*, 32(11), 1–4. https://doi.org/10.1029/2005GL022912

Tsai, W. P., Feng, D., Pan, M., Beck, H., Lawson, K., Yang, Y., et al. (2021). From calibration to parameter learning: Harnessing the scaling effects of big data in geoscientific modeling. *Nature Communications*, 12(1), 1–13. https://doi.org/10.1038/s41467-021-26107-z

Van Der Knijff, J., Younis, J., & De Roo, A. (2010). Lisflood: A gis-based distributed model for river basin scale water balance and flood simulation. *International Journal of Geographical Information Science*, 24(2), 189–212. https://doi.org/10.1080/13658810802549154

Warren, I., & Bach, H. (1992). Mike 21: A modelling system for estuaries, coastal waters and seas. *Environmental Software*, 7(4), 229–240. https://doi.org/10.1016/0266‐9838(92)90006‐p

Wilson, G., & Ozkan-Haller, H. T. (2012). Ensemble-based data assimilation for estimation of river depths. *Journal of Atmospheric and Oceanic Technology*, 29(10), 1558–1568. https://doi.org/10.1175/JTECH-D-12-00014.1

Xu, K., & Darve, E. (2020). Adcme: Learning spatially-varying physical fields using deep neural networks. Retrieved from http://arxiv.org/abs/2011.11955

Zhou, Z., & Tartakovsky, D. M. (2021). Markov chain Monte Carlo with neural network surrogates: Application to contaminant source identification. *Stochastic Environmental Research and Risk Assessment*, 35(3), 639–651. https://doi.org/10.1007/S00477-020-01888-9

Zhu, W., Xu, K., Darve, E., & Beroza, G. C. (2021). A general approach to seismic inversion with automatic differentiation. *Computers and Geosciences*, 151, 104751. https://doi.org/10.1016/J.CAGEO.2021.104751