Neural network to infer bed topography from velocity field: U-net architecture on huge experimental data

Mehrdad Kiani-Oshtorjani[1], **Christophe Ancey**[1]

[1]École Polytechnique Fédérale de Lausanne, Écublens 1015 Lausanne, Switzerland

Key Points:

- Entropy-based models applied to gravel-bed flumes could provide cross-sectional velocity fields,
- First attempt to use the U-net model for bathymetry inference have been done in this work,
- The U-net could provide satisfactory bathymetry inference from the velocity fields.

Corresponding author: Mehrdad Kiani Oshtorjani, `Mehrdad.Kiani@epfl.ch`

Abstract
Measuring bathymetry has always been a major scientific and technological challenge. In this work, we used a deep learning technique for inferring bathymetry from the depth-averaged velocity field. The training of the neural network is based on 5742 laboratory data using a gravel-bed flume and reconstructed velocity fields (namely, the topographies were obtained from real-world experiments and the velocity fields were estimated using a statistical model). To examine the predictive power of the proposed neural network model for bathymetry inference, we applied the model to flume experiments, numerical simulation, and field data. Results show the model properly estimates topography, leading to a model for riverine bathymetry estimation with a 26.3% maximum relative error for the case study (confluence of the Kaskaskia River with the Copper Slough in east-central Illinois state, USA). The dataset and the codes are attached to the present paper.

1 Introduction

Imaging riverine bathymetry is fraught with difficulty. Among the available techniques in riverine bathymetry, a typical example is provided by the use of airborne bathymetric LiDAR systems (Marcus, 2002; Wozencraft & Millar, 2005). While these techniques do provide direct access to bathymetry, it is time consuming and costly. Other examples include the use of indirect bathymetry techniques, which usually involve solving an inverse problem relating flow depth to surface elevation (Durand et al., 2008), surface velocity (Emery et al., 2010), and thermal imagery (Puleo et al., 2012).

Inferring bathymetry from surface flow data is predicated on the assumption of a strong causal relationship between the two (Smith & McLean, 1984). This assumption is often true, especially in shallow flows, when vertical velocity is low compared to the streamwise velocity component. Employing surface velocity data is an alternative to estimate bathymetry, thanks to its affordability and sensitivity to river depth (Landon et al., 2014; Wilson & Özkan-Haller, 2012; Ghorbanidehno et al., 2021). There are several techniques for estimating surface velocity, including the use of drifter GPS recordings (Honnorat et al., 2010; Landon et al., 2014). Landon et al. (2014) investigated whether drifters' trajectories are sensitive enough to bottom topography to allow for depth determination. They successfully extracted river bathymetry using velocity field measurements collected from drifter GPS records in an ensemble-based data assimilation approach. Estimated bathymetry based on this technique on a shallow braided and deep meandering reach of the Kootenai River in Idaho in the United State of America (USA) were more accurate than the previous estimations. Wilson and Özkan-Haller (2012) applied this technique to one dimensional (1D) channel, and for two real-world reaches, namely, the Snohomish River, Washington and Kootenai River, Idaho, in the USA. The main difference with Landon et al. (2014) was this: they used depth-averaged velocities (based on numerical solutions to the shallow water equations), and a least-square method to minimize a cost function that combines known information, and measured data. By using a state augmentation technique, the measured variable (velocity) is connected to the unknown parameter (bathymetry), providing a model for deep water bathymetry estimation (depth in the 3–10 m range).

In recent years, deep learning has become one of the most powerful tools for overcoming some deterministic approaches limitations. It can be used to image bed topography from surface flow data. Due to their ability to identify patterns or trends in data, neural networks are becoming increasingly popular in geophysics and hydraulics (Yu & Ma, 2021). For instance, Ghorbanidehno et al. (2021) used the neural network technique to obtain bed topography based on the depth-averaged flow velocity field, using limited labeled data. In order to reduce network size, the authors combined a fully connected deep neural network with principal component analysis (PCA). Usually, training the net-

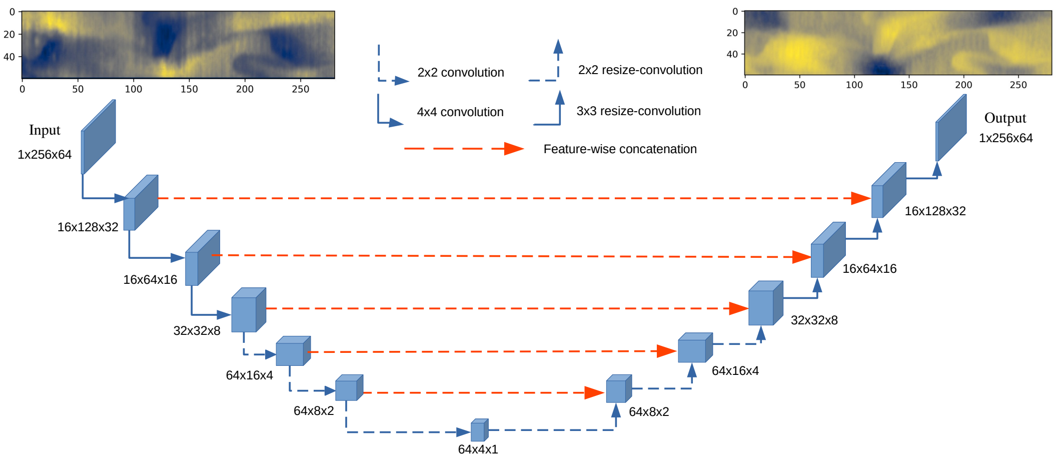


Figure 1. A diagram showing the input and output fields of the neural network. Convolutional layers are shown in blue, and skip connections are shown in orange.

work needs a huge amount of data to avoid the curse of dimensionality, which implies that the data occupy less and less of the data space as the data space moves from lower to higher dimensions. The volume of this space grows so fast that the data cannot keep up and thus become sparse—the sparsity problem is a major statistical significance issue. To enhance the training dataset size, Ghorbanidehno et al. (2021) divided the river's entire domain into small subdomains. As a result, each river profile provided several hundred training samples rather than just one.

The training of neural network for bathymetry estimation could be done by solving the shallow water equations and using the resulting solutions. A typical example is provided by Liu et al. (2022), who employed shared-encoders and separate-decoders, where bathymetry's input image is encoded and then decoded to three outputs, namely, the flow's longitudinal and transverse depth-averaged velocity components, and the water surface elevation. Two-dimensional (2D) simulations using randomly generated input bathymetry data were used to generate the training data.

Recent advancements in measurement techniques have led to the availability of high-resolution and low-cost surface velocity fields. These techniques include the use of Lagrangian drifters (Honnorat et al., 2010), large-scale particle image velocimetry (PIV) (Bradley et al., 2002; Lewis & Rhoads, 2015), and synthetic aperture radar (SAR) (Biondi et al., 2020). This work intends to predict bathymetry by using indirect measurements, i.e., the velocity field through a convolutional neural network. The neural network training is based on a U-net architecture, used for the first time for segmentation problems (Ronneberger et al., 2015), and to our knowledge, this is the first attempt to use the U-net model for bathymetry inversion. To this end, we used an experimental dataset made of 5742 data. It was not realistic to measure the velocity field, but we could estimate it using entropy-based models. A U-net neural network model is implemented in PyTorch framework (`https://pytorch.org`). Fig. 1 shows the model's architecture. In this network, the encoder and decoder are connected by skip connections - by contrast with regular convolutional networks. By doing so, we ensure we actually lose no information during the feature extraction process. In order to develop a well-trained neural network model, we need to collect a large amount of data. Due to the size of the experimental dataset used in this study, we had no problems with dataset size. As an alternative, if we were dealing with a small dataset, we could divide the experiment domain into smaller regions, thereby, we could increase the amount of data.

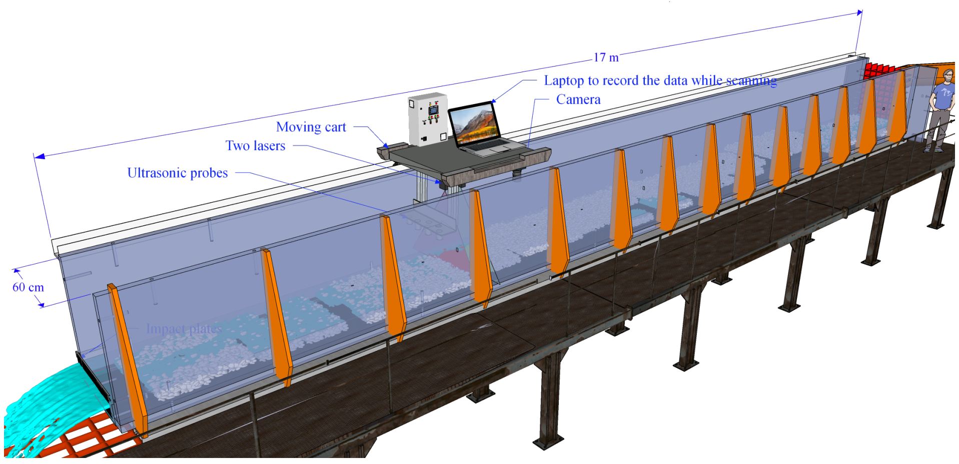


Figure 2. Gravel-bed flume illustration which all the dataset is gathered based on the experiments performed on this channel. It has 17 m length but the useful length (removing 1.5 m from each side) is 14 m.

In this work, training the convolutional neural network model to infer bathymetry is based on bed topography scans and velocity field estimates. Solving the governing partial differential equations will be bypassed and the solutions infered solely by convolutional neural network that is trained based on a huge experimental dataset. Closely related to our work is Ghorbanidehno et al. (2021)'s PCA-DNN framework, which combined the traditional fully connected deep learning method and principal component analysis. They trained the deep neural network model based on field data, whereas we focused on a convolutional neural network; besides, our dataset is produced in the laboratory. Our trained model is suitable for gravel-bed rivers whereas their model is suitable for deep rivers. All of the training networks in this work has been done using Colaboratory on GPU: Nvidia Tesla P100-PCIE-16GB.

This paper is structured as follows: The methodology for generating the dataset based on experiments and analytical derivation is explained in section 2. Moreover, the U-net network architecture is described and the basic parameters, regularization and hyperparameters - such as dropout, learning rate, weight decay, and training dataset size - are studied in that section. In section 3, the model's performance and accuracy has been studied. In addition, the neural network model's predictive power has been studied by applying it to flume experiments, the numerical simulation of a gravel-bed flume, and field data performed at the confluence of the Kaskaskia River and Copper Slough in central-east Illinois state, USA. Sections 4, and 5 discuss and summarize our achievements and future work, respectively.

2 Methodology

2.1 Data generation

The dataset is based on the experiments conducted at LHE-EPFL (Dhont, 2017). Table. 1 represents the input parameters and each experiment's duration. It consists of three long experiments. Experiments were carried out in a tilted flume at an angle of $S = 1.6\%$ and 1.7%, a length of $L = 17$ m and a width of $w = 60$ cm, as depicted in Fig. 2. The useful length of the flume is 14 m because of technical limitations (1.5 m from each side is ignored). The flume bed was made of natural gravel with a height of 31.5

cm at the beginning of the experiments. Sediments mean diameter was $d = 5.5$ mm, with a 1.2 mm standard deviation, and density of $\rho_s = 2660$ kg/m^3. The water discharge and sediment feeding rates at the flume inlet were set to 15 L/s and 2.5, 5, and 7 g/s, respectively. Each experiment lasted for hundreds of hours. At the beginning of each experiment, the bed surface was flattened. These long-term experiments are conducted via short-term experiments that last 8 hours. To prevent the destruction of the bed topography, these short-term experiments are lunched with very low flow discharge. During the experiment, water discharge and sediment feeding rate were kept constant for 8 hours. Experiment 1 included 1566, experiment 2, and 3 consisted of 741, and 3435 scans, respectively (in total 5742 scans, each consisting of bed topography and flow depth data). Based on the entropy-based models, we computed the depth-averaged velocity field using the flow depth data and knowing the constant flow discharge as a constraint. In the next following sections, the details of entropy-based models to inferring the velocity fields will be explained.

Table 1. Input parameters of the experiments.

	Exp. 1	Exp. 2	Exp. 3
Flow rate (L/s)	15	15	15
Flume slope (%)	1.6	1.7	1.6
Sediment feed rate (gr/s)	2.5	7.5	5.0
Duration (h)	250	556	118

During an experiment, measurements of the bed topography and flow depth are recorded with a fine resolution. The bed topography is measured by laser-sheet imaging technique i.e., using two angled lasers. Those lasers are mounted on an automated moving cart. By calibrating the distance between two angled lasers projected on the bed, the bed topography can be induced with a resolution of 60 × 281 pixels. Every 10 minutes, the cart programmed to scan the bed proceeds, using MatLab. Each scan took about 145 seconds and covered 14 m × 60 cm. During each scan, the bed topography by lasers, and flow depth by ultrasonic probes are measured. The ultrasonic probes determine flow height, a sound pulse is emitted and the travelling time between the sensor and the object can be used to infer flow depth (see Fig. 2). The sediment feeding system moves the sediments from the hopper by a conveyor belt into the inlet of the flume and feeds into the channel through a pin board that distributes the gravel along the width. The sediment feed rate was controlled by controlling the speed of a rotating cylinder that clogs up the hopper outlet. The most frequent mode happening in the flume using the aforementioned flow condition is alternate bars. In Fig. 3, the bed topography and alternate bars appearance are illustrated based on the third experimental data at $t = 200$ minutes.

Due to erosion, deposition, and transport processes, the bed topography changes over time during the experiments. In turn, the hydraulic conditions in the system are affected by the bed morphology, as a result of a feedback loop driven by the movement of particles (Griffiths, 1993). Based on the Froude number $Fr = v/\sqrt{gh}$ (where g is the gravitational acceleration, v is flow velocity and obtained by particle image velocimetry using tracking light polystyrene balls, and h is flow depth measured by ultrasonic probes), the flow regime was turbulent and super-critical with a Froude number more than 1. As a result of the highly variable bed topography and hydraulic conditions in alternate bar systems, calculating the shear stress is not straightforward. Since the flow depth is uniform at the beginning of each experiment, thus the streamwise bed shear stress can be measured by $\tau = \rho g R_h S$ where $R_h = hw/(2h + w)$ is the hydraulic radius, and ρ is

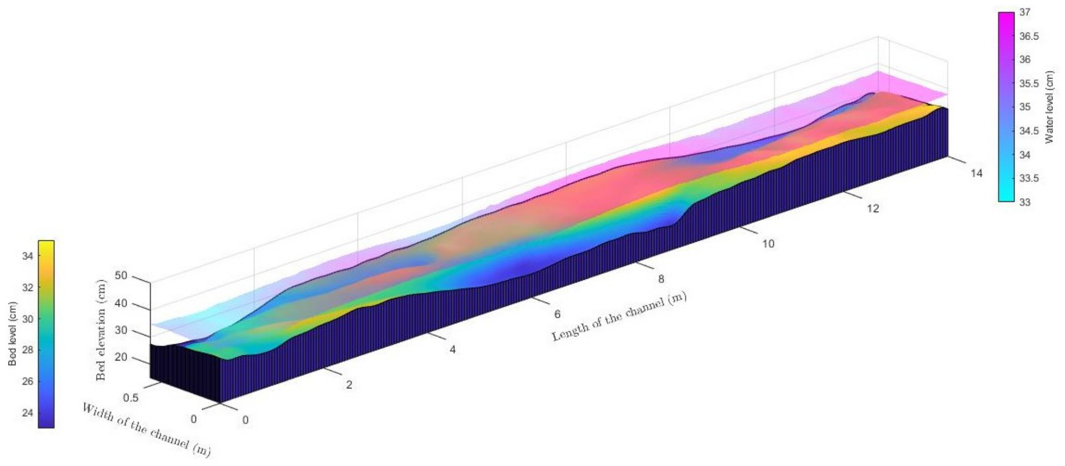


Figure 3. An example of experimental topography using the gravel-bed flume. The bed and flow heights are measured from the bottom of the flume. The initial height of the bed is at 31.5 cm.

fluid density. The shear stress is estimated by computing its average value along longitude bed profiles (Venditti et al., 2012). The value of shear stress in our experiments varies between 4.89 to 10.75 kg/(ms^2). The Shields number and the shear velocity can be obtained by $\tau^* = R_h S/R/d$, and $u^* = \sqrt{\tau/\rho} = \sqrt{gR_h S}$, respectively where $R = (\rho_s - \rho)/\rho$. The shear velocity varies in a range of $0.07 \le u^*(m/s) \le 0.10$ and the Shields number changes between $0.05 \le \tau^* \le 0.11$ among all experimental data.

In order to have a suitable input/output image size for training the neural network, we applied a bicubic interpolation over 4×4 pixels neighborhood to the flow depth and bathymetry data to change the size of the data i.e, bed topography and velocity fields from 60×281 into 64×256 pixels. Then, the dataset was divided into three subsets: training, validation, and test dataset. The total number of samples was distributed this way: 80% training data, and 20% validation data. Moreover, the training data was again divided to 90% as the training and 10% as the test dataset. Training, and validation data will be used to fit the models, and to estimate the prediction error for model selection, such as hyperparameter tuning, respectively, and finally a test dataset will be used to assess the generalization error of the selected model. The dataset composed of 5742 pairwise (bed topography and velocity field) grayscale images and, therefore, the training, validation, and test datasets consist of 4133, 1149, and 460 data, respectively. The dataset are published attached to this work on `Kaggle.com/MehrdadKianiOsh/bedtopo_vel_fields`. A total of 11484 fields are included in the published dataset, including both bed topography and depth-averaged velocity data.

2.2 Entropy-based velocity profile

The entropy concept is originated from thermodynamics. Entropy is defined as a measure of a system's randomness or disorder. Shannon developed entropy's mathematical foundation and connected it to information (Singh, 2016). To obtain the two-dimensional and one-dimensional velocity profile, it is necessary to maximize the entropy of the velocity distribution, in order to obtain the least biased velocity probability density function. It is based on the Jaynes (1957) principle of maximum entropy (POME) that entropy should be maximized. According to Jaynes (1957), any system in the equilibrium state attempts to maximize its entropy, subject to given constraints. The entropy of a flume/river must reach its maximum value when it reaches a dynamic (or quasi-dynamic)

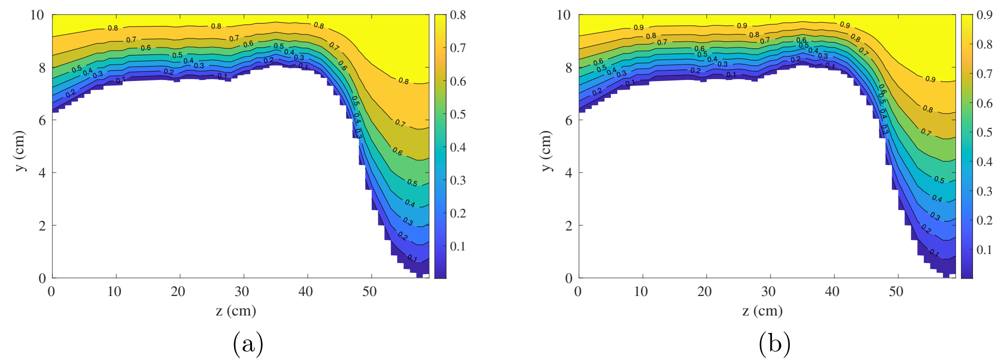


Figure 4. Velocity plot based on (a) Shannon-entropy, and (b) Tsallis-entropy for a cross section in the middle of the channel based on Exp. 3 at $t = 200$ minutes. The border between the white and blue regions indicates the bed topography for one cross section.

equilibrium (Singh et al., 2003). At any location where maximum velocity occurs, the 2D velocity distribution based on entropy theory should be valid (Singh, 2016).

As aforementioned, during each experiment, bed topography and flow depth are measured locally with high resolution. Since the flow discharge is fixed to a constant 15 L/s value during all the experiments, the constraint in the entropy-based models is constant flow discharge. The velocity profile can be obtained locally by maximizing entropy. Estimating flow velocity can be achieved using a number of techniques. Two entropic principles are known as Shannon and Tsallis (this entropy is a generalization of the Shannon entropy) that are often applied to river discharge assessment (Singh, 2016). Both these principles connect the maximum flow velocity at a vertical axis of the flow area to the cross-sectional mean flow velocity.

2.2.1 Shannon entropy-based method

Chiu and colleagues (Chiu, 1987, 1988, 1989) introduced the use of Shannon entropy theory to the field of hydraulic engineering. By utilizing an entropic parameter, they established a linear relation between the maximum flow velocity and the mean flow velocity over a cross-section. Building upon Chiu's work, Moramarco et al. (2004) proposed a simplified version of Chiu's entropy-based velocity distribution equation that can calculate the 2D velocity distribution, using only the maximum velocity and bathymetry information. This model does not require parameter calibration or the isovel equation, and only requires knowledge of the maximum velocity and its position. Generally, maximum flow velocity occurs at the water's surface; however, the phenomenon of the dip-phenomenon may occur, in which maximum flow velocity occurs below the surface, due to secondary currents. When $w/h > 3.5$, the maximum velocity occurs on the flow surface (Song & Graf, 1996) which is true of our experiments. The velocity equation can be expressed as follows (Moramarco et al., 2004)

$$u = \frac{u_{max}}{M} \log\left[1 + \left(e^{M} - 1\right) \frac{y}{h - D} \exp\left(1 - \frac{y}{h - D}\right)\right] \tag{1}$$

The equation for estimating the longitudinal velocity (u) along the vertical axis includes several variables, namely, the maximum flow velocity, and its depth from flow's surface is represented by u_{max} and D, respectively. Shannon's entropy parameter is denoted as M, and y represents the location of the velocity measurement point from the bottom of the channel. To determine the constant value of the entropy parameter, the following equation can be used

$$\frac{u_m}{u_{max}} = \frac{e^M}{e^M - 1} - \frac{1}{M} \tag{2}$$

The mean velocity over a cross section u_m can be identified if we know the flow discharge and the section area, while u_{max} is determined iteratively (by knowing that the maximum velocity happening on the flow surface and assuming the maximum velocity in the first iteration and correcting the maximum velocity iteratively by calculating the flow discharge and comparing with the ground-truth). The contour plot of the calculated velocity profile is plotted on Fig. 4(a). However, for the sake of comparison, Tsallis entropy-based method have been applied to the same cross section (the methodology is explained in Appendix B). The velocity profile based on Tsallis entropy is plotted in Fig. 4(b). One can see that the velocity gets close to 1 m/s on the surface flow, which coincides with the measured surface velocity by tracking polystyrene balls travelling along the entire flume length (Dhont, 2017). By comparing two velocity fields in Fig. 4(a) and Fig. 4(b), there is no such difference in using each entropy models. We therefore decided to use Shannon's entropy to calculate the velocity fields in all of our experiments.

2.3 Neural network

We use the U-net architecture for our neural network model. In the main journal paper where the U-net has been introduced by Ronneberger et al. (2015), the input and output images were of different sizes and used to solve the segmentation problem. Since our input/output are of the same size, we reworked the model by changing the convolution function parameters and using the model as a regression problem. It has already been mentioned that the U-net network is divided into two parts. Firstly, a standard convolutional neural network architecture is used to perform the contracting process. Leaky ReLU activation units are followed by multiple convolutions with padding in the contracting path. The same structure is repeated several times. One of U-nets' key characteristic lies in the expansive path i.e., the second path. Using transposed convolution, each stage in the expansive path upsampling the feature map. Afterwards, we concatenate the upsampled feature map with the corresponding layer from the contracting path. Therefore, we obtain a U-shaped network and, perhaps most importantly, contextual information is propagated along the network, enabling a proper reconstruction of context.

The implementation is based on Pytorch deep learning framework (`https://pytorch.org`). Table A1 in Appendix A shows the structure of the U-net used in all our tests with details of the different layers. The schematic of our neural network architecture can be seen in Fig. 1, in which a fully convolutional U-net is used. This is a famous architecture that uses a number of convolutions at various spatial resolutions. This network differs mainly from a regular convolution network in that skip connections are used from encoder to decoder parts. By doing this, the network can use fine-grained details learned in the encoder to reconstruct an image in the decoder. Using it as a whole is the only way to make this devicer work. For example, if we want to use the decoder as a standalone component, it does not work. No pooling is used. We used strides and transposed convolutions instead (they must be symmetric in the decoder path, that is, have an uneven kernel size).

2.3.1 Basic parameters

In order to tune the training hyperparameters such as learning rate, learning rate decay, and normalization, these hyperparameters need be evaluated using the baseline architecture. The learning rate is a tuning parameter of the optimizer function that determines the step size that the optimizer takes at each iteration while moving toward a

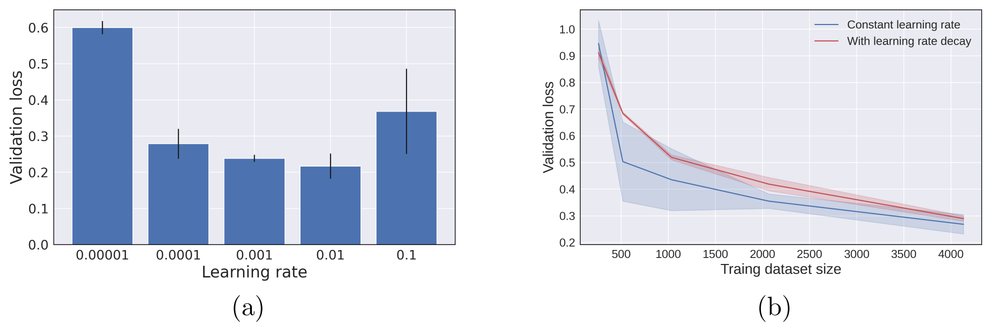


Figure 5. (a) Effect of different constant learning rates in terms of validation loss, (b) testing models based on validation loss with and without learning rate decay (variable amounts of training data).

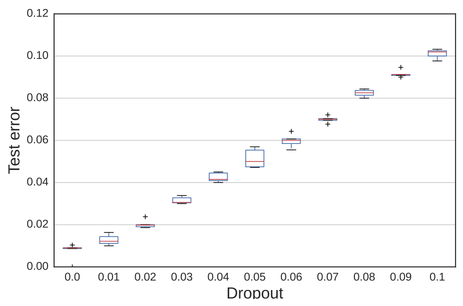
loss function minimum and to update a neural network's weights using a gradient computed from one data mini-batch. Fig. 5(a) shows the validation loss using different learning rates. As can be seen, it is clear that the lowest and largest learning rates overshoot and have difficulty converging, considering our problem, we have the best learning rate with a learning rate of 10^{-2}, and 10^{-3}.

With gradient-based optimization algorithms, saddle points can be escaped, and training takes longer time since surface around such points are flatter and gradients close to zero (Goodfellow et al., 2016). Therefore, rather than using a fixed value for learning rate, it could be decreased over time. If training no longer improves our loss, the learning rate is changed, every iteration being based on some cyclic function f. The number of iterations in each cycle is fixed. By using this method, the learning rate is allowed to vary cyclically between reasonable boundaries. We can traverse saddle point plateaus more quickly when we increase our learning rate, since we are less likely to get stuck in undesirable states like saddle points. We have performed a test: in it the learning rate decays by a gamma factor in every epoch that is set to be 0.99 i.e., $lr_{epoch} = \gamma \times lr_{epoch-1}$. Fig. 5(b) shows based on this strategy. We found out that the learning rate decay does not help the improvement of the model based on validation loss comparison while it helps decrease diversity around the mean value. By decreasing the learning rate with a $\gamma = 0.99$ in each iteration during training - since it reduces the variance in performance -, it helps stabilize results, but it does not reduce validation loss. The influence is minimal, if other parameters are chosen appropriately.

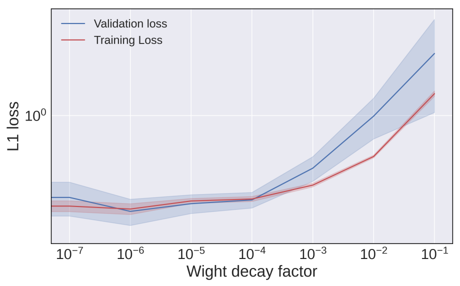
Based on Fig. 5(a) and Fig. 5(b) tests, we used Adam optimizer with a fixed learning rate of 0.001 and betas=$[0.5, 0.999]$ for the rest of the training optimizations. In addition to a nonlinear activation function, each of the network section contains a convolutional layer and a batch normalization one. In the model, we included the possibility of normalizing batches. During training, it shifts and rescales according to the mean and variance estimated on the batch. The literature has proven that batch normalization makes the training process smoother. However, it requires a sufficiently large batch size, and our choice of batch size = 100 may be too restrictive. We did not use any pooling for the process.

2.3.2 Regularization and hyperparameters

Regularization is an essential subject for deep learning algorithms because it inhibits overfitting, that is, the model learns unnecessary details, noises or random fluctuations in the training data as main concepts, insofar as it negatively impacts the model's



(a)



(b)

Figure 6. (a) Dropout rates of a model with 142,689 weights are compared applied to all the training data samples, (b) training and validation performance on the weight decay factor.

performance. Dropout, weight decay, data augmentation, weight initialization, and early stopping are the most often used regularization techniques. The choice of initialization is highly influential on most algorithms when training deep models. Algorithm convergeability can be determined by the initial point, with some initial points being so unstable that numerical difficulties arise and the algorithm fails. The weight initialization of a neural network consists of setting its weights at small random values, which determine the starting point of the optimization when learning or training. In order to prevent layer activation outputs from exploding or vanishing gradients during training, an appropriate initialization of the weights is necessary and manages to achieve better performance. Here are the most widely used, among others: the He normalization (He et al., 2015) and Xavier normalization (Glorot & Bengio, 2010), when dealing with convolutional layers. We experimented with both initializations for our networks and the former seems to lead to slightly better performances than the latter. We therefore sticked to that one. Based on the He initialization method, initialization is based on a randomly generated number computed with a Gaussian probability distribution with a mean of zero and a standard deviation of $\sqrt{2/n}$, where n stands for the number of inputs to the node. We can go one step further by adding mechanisms specifically designed to facilitate the training such as Dropout (Srivastava et al., 2014). Dropout consists of randomly dropping out neurons in a layer, with a probability that corresponds to the dropout coefficient. We implemented the possibility to include it in the model. Here is this technique's main advantage: it prevents overfitting as the neurons of a layer become less dependent on particular inputs. Dropout is not recommended in our work. In Fig. 6(a), the influence of the Dropout is studied, which has a negative effect on the test error. Therefore, the Dropout is not recommended for this study and it is considered as zero for the rest of the training. This might be the reason: dropout is usually unnecessary when the network is small compared to the dataset. By adding this regularization, it will worsen performance if the model capacity is already low.

On the other hand, using data augmentation to increase the amount of data is difficult in a physical application such as this work because, for a given input configuration, each solution is unique. Therefore, we did not use any data augmentation methods in this work. In order to further improve the models' performance, we can also use the weight decay method. As this study is centered on convolutional neural networks, which are inherently less prone to overfitting than fully connected neural networks, the weight decay strategy is especially important for them. This approach consists in adding a penalty to the model based on the amplitude of its weights, in order to limit overfitting. The model will all the more be penalized as the values of the network connections increase. This strategy is based on the principle that large weights in a neural network can cause more variance at output and prevent the model from generalizing correctly.

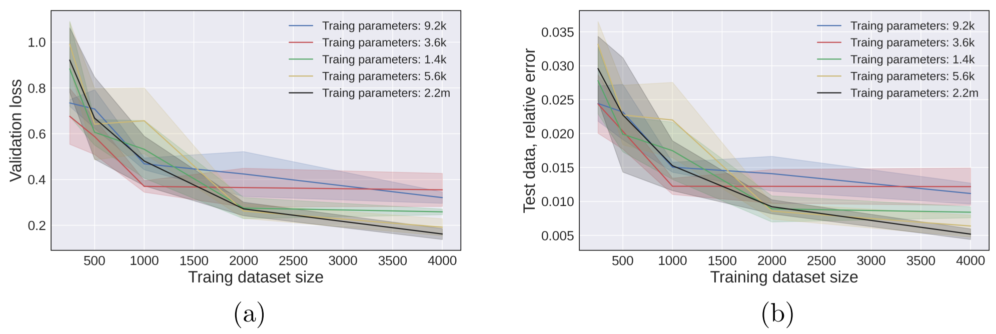


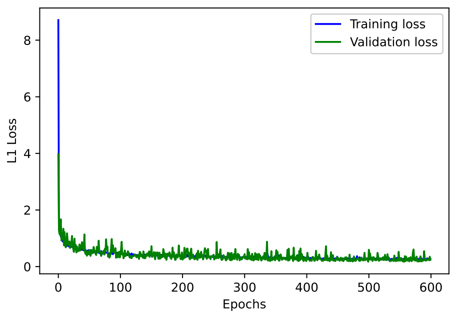
Figure 7. (a) Validation loss (L1) for models with different sizes and quantities of training data, (b) influence of different model sizes and amounts of training data on testing accuracy relative error.

The penalty forces weights down, and permits to obtain a less flexible network that is less specialized in the data used for training. This penalty is defined as follows $L_{weight\_decay} = \lambda \sum_i \omega_i^2$ where L_{weight_decay} is the penalty associated with weight decay, ω_i is the i$-th$ weight in the network and λ is a positive coefficient that affects the importance of the penalty. This parameter is yet to be determined, so we carry out training with several values of λ as presented in Fig. 6(b) in which the validation and training loss are plotted as a function of the weight decay factor. Losses for the weight decay factor larger than 10^{-4} start to increase while the losses for the lower weight decay reach a plateau. The training time using different weight decay factor does not change dramatically, therefore the best weight decay could be chosen by just paying attention to the losses in which 10^{-6} is the best for the rest of the training.

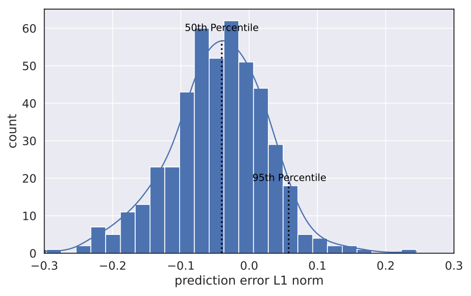
3 Results and performance

3.1 Model performance

The network in this work consists of a series of convolutional blocks and is fully convolutional with 12 layers. All blocks structures are similar: activation, convolution, and batch normalization. An upsampling followed by a convolution rather than a transpose typical convolution is used instead. Convolutional blocks in Appendix A is parameterized by channel factor, kernel size, stride, and padding. In the following training runs, the number of feature maps in the U-net convolutional layers are scaled from 250 to 5742 samples. The total amount of weights is modified by varying the number of feature maps. With a $\times 2$ increase in input channels, overall weights will quadruple as a result (there is a linear change in biases), because $K = C_{in} \times C_{out}$ where K be the size of the kernel tensor, C_{in} be the number of input channels, and C_{out} be the number of output channels. Fig. 7 displays the accuracy results for five distinct network sizes. An "EXPO" variable is defined in order to control the exponent for feature maps in the neural network model. The size of a network can be scaled directly by this parameter. For example, 3 gives a network with 9.2k parameters. It is relatively small for a generative neural network with $1 \times 64 \times 256 = 16$k outputs, but it allows for faster training time and prevents overfitting in view of the relatively small dataset we are working with. There are 142689 trainable parameters in this network, with an exponent of 3. This is a crucial number to keep in mind when training neural networks. It is easy to change and increase the EXPO parameter, and end up with a network with millions of parameters, then it is highly probable we will be faced with all kinds of convergence and overfitting issues. To avoid the dimensionality curse, the number of parameters must match the amount of train-



(a)



(b)

Figure 8. (a) Training and validation losses for learning rate 0.001, (b) histogram plot of the $L1$ norms of prediction errors (460 test cases).

ing data, as well as scale with the network's depth. The exact relationship between these three depends on the problem under consideration.

In Fig. 7(a), different models perform differently for 1149 velocity fields from the test dataset that were not seen during training. Using the same distribution used for generating training data, velocity fields were randomly sampled. By using these data, we can determine the generalization capacity of the trained models, when applied to new velocity fields. A mean relative error is shown in this graph instead of the $L1$ loss function which have been used for the validation data (relative errors and $L1$ loss behavior are similar here). Relative error is calculated using $err_r = 1/n \sum |\hat{y}_b - y_b|/y_b$, with n representing the number of samples, y_b referring to the function in question, i.e., bed elevation, and $\hat{y}_b$ represents the estimation of bed elevation, calculated based on the neural network model. For evaluating the models, we found the average relative error to be a good metric, since it accounts for all outputs and directly indicates the inferred solutions accuracy. Besides, it provides an indication on the estimated solutions relative accuracy, whereas the $L1$ metric represents an estimate of averaged differences. Therefore, both metrics are crucial to assess how accurate the trained models are overall.

Curves in Fig. 7(b) show the relative errors for various model sizes and training dataset size. There is a small difference between the different model sizes for a specific number of training samples, but the fall-off in error for larger amounts of training data appears to be similar. We can see that using the largest number of training dataset size i.e., 4311 sample data, the difference between test errors for models with the smallest and largest amount of training parameters is 0.2 and 0.005, respectively. This leads to a 2.6% relative error on average for the latter network. Fig. 8(a) shows the neural network is trained well without overfitting. After 600 epochs, the validation and training losses should have decreased from an initial value of around 8.9 to 0.02 based on the standard settings. Long-term trends in the training can be identified by doing this. It can be difficult to determine if the overall trend of noisy numbers (the losses) in a command line log is going up or down, but this can be seen in the visualization shown in Fig. 8(a). It is visually easy to see the loss curves trend down for 100 epochs and, afterwards, the curve flattens out. As we move towards the end by increasing the number of epochs, the validation loss is still decreasing slowly, and most importantly, it is not increasing. A divergent validation loss from the training loss would indicate overfitting, something we should avoid. The graph illustrates the models do not exhibit overfitting over time and reach to stable levels of validation and training losses.

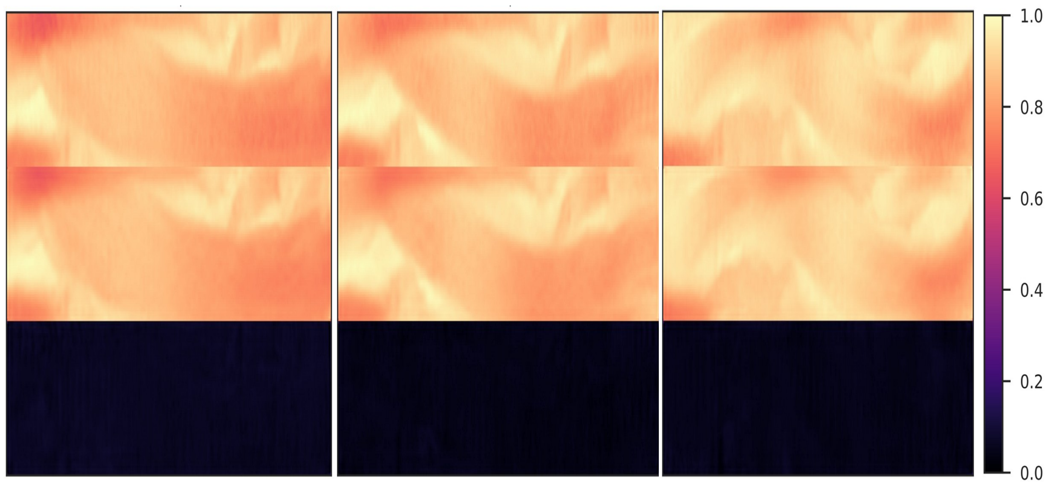


Figure 9. Qualitative result of the U-net: the first row is the bed topography fields from the test dataset (ground truth). The second row are the images that are predicted by the U-net. The third row are the error images (ground truth - prediction).

To study the neural network model performances, in which we have tuned the parameters in the previous sections on a larger number of training and validation datasets, we perform an error measurement for all the test dataset. Fig. 8(b) shows the distribution of $L1$ error. The 50th and 95th percentile are -0.041, and 0.058 with a mean, and maximum error of -0.045 and 0.32, respectively. Most examples lie close to zero and the maximum error is 3.2 mm while the bed topography changes in an interval of 18.19 to 37.02 cm, meaning in a 18.83 cm range. In Fig. 7(b), it has been shown that the best model has a 0.5% relative error which means the model is well trained and provides accurate predictions.

3.2 Model's predictions

Following the establishment of a stable training setup in the preceding section, the best trained network's prediction capability may now be investigated further. We explored how the size of the training dataset affects accuracy for the validation dataset, and generalization accuracy for the test dataset, using a database with 5742 input data and target solutions developed as mentioned in the previous sections. Additionally, we determined how accuracy varies with the number of weights utilized in the model (degrees of freedom in a neural network). Afterwards, by selecting the best model, namely, having 1.4 k parameters and with no dropout, learning rate of 10^{-3}, and choosing 10^{-6} as the weight decay factor, and using the whole dataset, i.e., 4133 data points for the training, the final neural network is established and used for studying accuracy and performance on different case scenarios.

3.2.1 Model's predictions based on experiments

A part of the dataset (460 data points) is kept for testing the neural network model's prediction power. The velocity fields in the test dataset feed into the neural network and the prediction is examined by comparing with the ground truth (experiments). The accuracy of the model is shown in Fig. 9 so that each column represents one of the bathymetry in the test dataset. The first row is the ground truth of the bed topography and the sec-

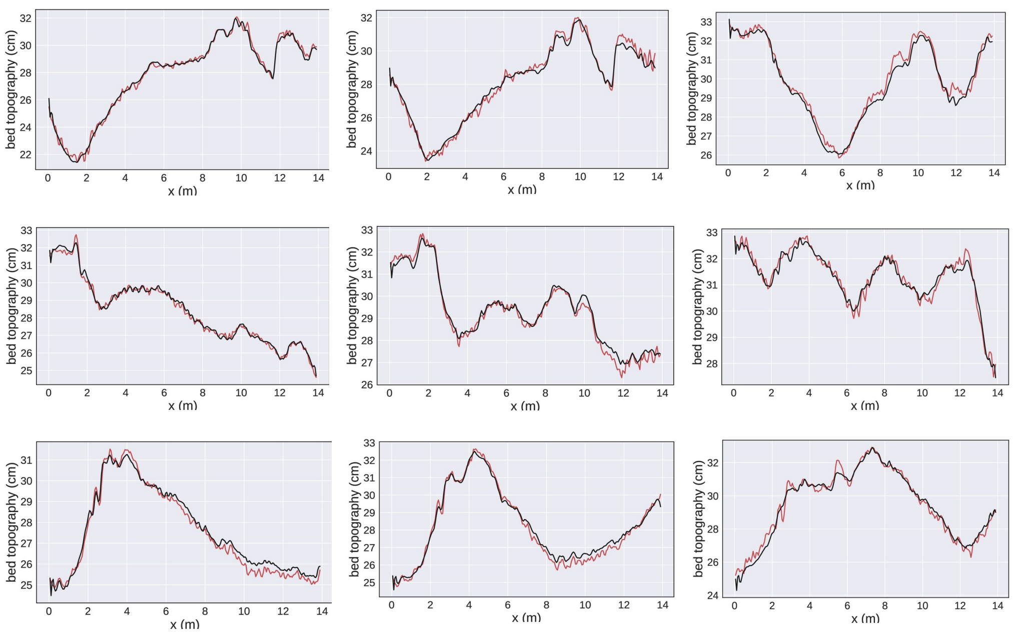


Figure 10. Each column belongs to one experiment. The first row is the profile of the bed topography along the channel length at $z = 5$ cm from the down wall, the second row is the profile at $z = 30$ cm, and the last row is the bed topography at $z = 55$ cm. Ground truth is shown by red and prediction by black color.

ond row shows the predicted bathymetry, using the neural network model. The difference between aforementioned topography fields (ground truth - prediction) are presented in the third row. The mean error among all the test dataset is 0.45 cm, while the topography varies in a range of 18.8 cm.

To quantify the model's prediction further, the topography over a longitudinal axis along the flow direction is compared in Fig. 10 with the neural network model's predictions. The first to third columns in Fig. 10 belong to the first to third columns in Fig. 9, respectively. The longitudinal axes are located in $z = 5$, 30, and 55 cm from the down side wall. The black and red solid lines represent the prediction and the ground truth (experimental data), respectively. As can be seen, the curves variations are finely reproduced and values coincide very well. This allows to conclude that hyperparameters related to the training and the architecture of the neural network model are chosen properly and provide sufficient accuracy. In addition to the validation of the model based on the test dataset, the numerical simulation performed and field data measurements are used to compare with the model's prediction in the next subsections.

3.2.2 Model's predictions based on numerical simulation

In subsection 3.2.1, the model is applied to the test dataset coming from experiments conducted at LHE-EPFL, and the model provided accurate predictions. Since the model is trained on the same dataset (the training part of dataset), therefore the model prediction using the test dataset in the previous subsection just shows us the model is well trained and that, if someone applied the model to gravel-bed laboratory channel, they can get very accurate results. An important question would be about the model's accuracy on the other applications. In order to further study the model's prediction power,

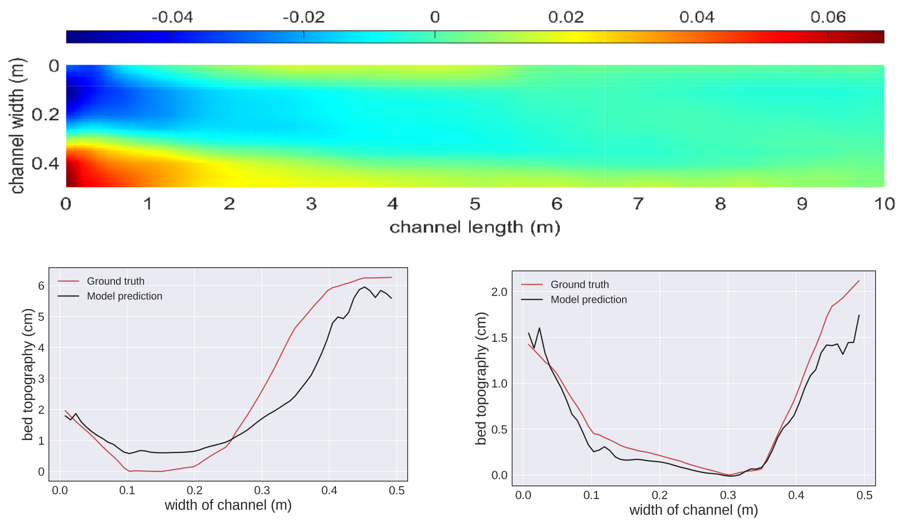


Figure 11. Up: Bed topography based on simulation using Iber, down: bathymetry comparison of the Iber and neural network model. The red line is the Iber topography for a cross section at $x = 1.0$ m (down-left), and $x = 5.5$ m (down-right) and the black line represent the neural network model's prediction.

the model will be compared with the simulation performed based on Iber (Bladé et al., 2014).

A gravel-bed flume is simulated based on Iber. A flume with a $Q = 15$ L/s flow discharge, 1% slope, 10 m length and 0.5 m width, using a structured mesh grid with 100 elements in the flow direction and with 10 elements in the transversal direction has been simulated with final time of $t_f = 7200$ minutes. The particle diameter considers 7 mm in addition of using Manning coefficient of 0.025, and choosing Meyer-Peter and Müller (MPM) as the bedload model on the software. The inlet boundary condition is set to constant flow discharge and an open boundary condition for the output has been chosen. Afterwards, the velocity field and the topography at the end of simulation time i.e., $t = 7200$ minutes are extracted from Iber. The trained neural network model has been applied to the depth-averaged velocity field extracted from Iber and the topography prediction from the neural network model has been compared with the topography of Iber.

Top plot in Fig. 11 is the channel bathymetry based on numerical simulation and down plots are a comparison of the neural network prediction versus the bed profile based on Iber on a cross section at $x = 1.0$ m (left plot), and $x = 5.5$ m (right plot). Fig. 11 shows the neural network model's accurate prediction in comparison with the ground truth (bathymetry from Iber), in which prediction has a 25% maximum relative error. The mean absolute error in cross section $x = 1.0$ m is 0.75 cm, while at the cross section $x = 5.5$ m is 0.14 cm. However, gaining such mean absolute error is small in comparison of the range in which bathymetry changes. This validation shows that, as long as the flow satisfies the mass and momentum conservation laws, the trained model is robust and accurate to predict bathymetry. Despite finding out about the neural network model performance in laboratory experiments, and simulation, inquiring about model performance on the field data is a natural question and an intriguing subject to examine, which will be answered in the next part.

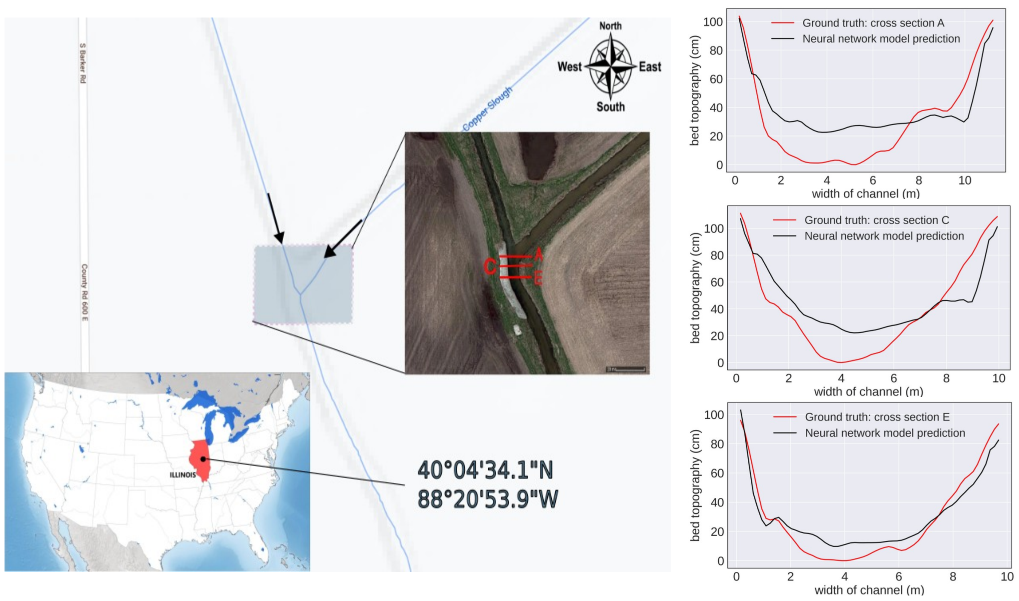


Figure 12. (left) Overview of the study site in east-central Illinois state, USA; confluence of the Kaskaskia River with the Copper Slough. The arrows show flow direction. The red lines represent the cross-sections where these measurements have been conducted. (right) comparison of the neural network prediction vs the field data measurements. The red and black lines represent the field measurements and neural network model prediction, respectively.

3.2.3 Model's predictions based on field data

Predicting the trained neural network model on the field data would be interesting to investigate. Thus, we performed model predictions on a river located in east central Illinois state in the United States of America (USA). Lewis and Rhoads (2018)'s work on confluence stream provides the velocity measurements using the acoustic Doppler velocimeter (ADV - Nortek Vectrino+) in addition to bed topography for several cross sections. Experiments were conducted at the confluence of the Kaskaskia River and Copper Slough (KRCS), positioned at 40°04'34.1"N, 88°20'53.9"W, as can be seen in Fig. 12. The width is about 20 meters at the confluence center and approximately 10 m downstream. ADVs mounted on a topset wading rod and placed at predetermined cross sections within the flow were used to determine the vertical velocity field. Three cross sections, namely, A, C, and E (see Fig. 12) are used for the purpose of bathymetry inference based on our neural network model. Sample volumes for ADV are 0.125 cm^3 and sampling frequency is 25 Hz, configured in laboratory mode. ADV sampling was carried out with a downward-looking probe 6 cm below the surface, to position the sampling volume as close to the surface as possible. $60-80$ samples 60 s in length are collected at three cross-sections during one measurement campaign. Cross-sectional ADV measurements were obtained the same day. All field campaigns resulted in only a few centimeters change in water surface elevation between the beginning and the end of the measurements, far less than the average flow depth. Cross-sectional velocity fields are fed into the neural network model. In order to use the trained model on the real-world river, having prior knowledge is necessary, namely knowing the river's maximum depth. The bathymetry prediction for the aforementioned cross-sections based on the neural network model are compared with the field data in Fig. 12's right-hand side. The ground truth (field data) is shown in red solid line while the prediction of the trained neural network model has been shown in black color.

The predictions' absolute errors in comparison with the measurements in cross section A, C, and E are 16.11, 15.23, and 7.75 cm, respectively and maximum relative errors are 26.31%, 24.39%, and 15.18%, respectively. However, the maximum relative error and the absolute error is still small compared to the approximately 100 cm depth of the river, while these errors could be decreased by adding a part of the field data to the training dataset, enhancing the number of training data, or using a more complicated model with more layers and training parameters.

4 Discussion

This work is concerned with imaging topography from the depth-averaged velocity field, which can be obtained more easily with a lower cost-rate than the bed topography direct measurement. The network connects velocity field and bathymetry. How the error in Fig. 8(b) can further be reduced, however, is naturally an important question. According to our tests, refer to Fig. 7(a), and Fig. 7(b), this requires a substantial increase in training datasets, and more complicated convolutional neural network models such as increasing the number of feature extractions. As Fig. 7 shows, merely increasing the size of the model and training data will not result in increments in accuracy. Therefore, different approaches and network architectures need be investigated. Assessing the model's systematic errors is a crucial aspect of the investigation. To this end, Fig. 9 presents a subset of inferred results obtained using the 1.4 k model trained on 4133 data samples. This investigation's purpose is to explore the extent to which model errors are systematic. The bottom row of Fig. 9 shows the magnitudes of the model's errors applied to flume experiments. In any one of these cases, the model is not completely off the error. The error distribution on the test dataset plotted in Fig. 8(b), while the mean value is very close to zero, i.e., the model is well-trained and the hyperparameters have been chosen properly.

The trained convolutional neural network model used the depth-averaged velocity fields, while field scientists usually measure surface velocities, for instance, by using particle image velocimetry method. Therefore, it was worth asking how much the error propagates when a field scientist feeds the surface velocity fields into the trained model in this work. The experiments (which we have used in this work to train the neural network) have been performed in a straight channel (no cross-stream secondary current), near steady state, with weak sediment transport. In these conditions, the velocity profile has been obtained by Song et al. (1994). Fig. 13 shows the flow velocity profile for uniform flow at various bed slope $0.25 < S(\%) < 1.5$, and discharge $30 < Q(L/s) < 130$ measured by Acoustic Doppler current profilers (ADVP). The log-law for the inner region ($y/h < 0.2$) is presented in red solid line and the Coles wake law could represent the velocity profile in the outer region ($y/h > 0.2$) in a blue solid line, and the black solid line is the averaged mean velocity. As one can see, by increasing the flow depth, the mean velocity deviates from the velocity profile, and the maximum relative error happens on the surface flow, with a 16.8% magnitude.

In order to know how this error was propagated in the model prediction, we have applied the trained convolutional neural network model to the field data and gained a 26.31% maximum relative error. Therefore, however, the training of the neural network model is based on depth-averaged velocity fields but it is possible to use surface velocity to infer bathymetry while accepting a relative error in the order of a couple dozen percentage.

Another concern about this work was about the amount of data needed to suffice for training the neural network. A test on the dataset size in Fig. 5(b) shows that 4133 data points was enough for this work's purpose, since the error does not significantly decrease after a 2500 dataset size. However, having 26.31% maximum relative error when the model applied to real rivers is good enough for real-world applications, however fur-

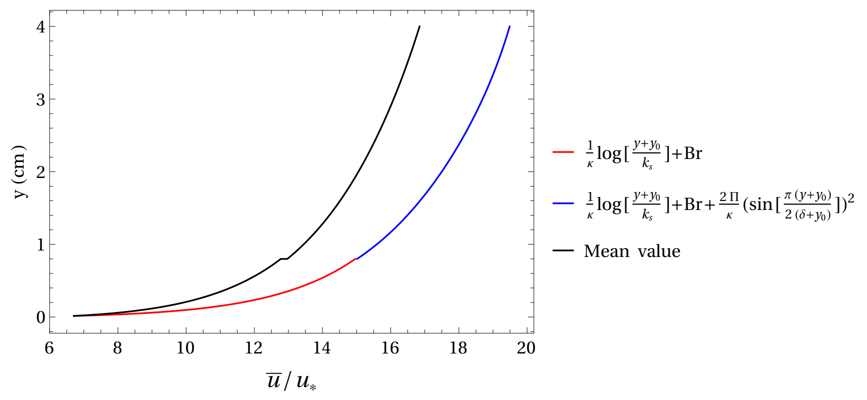


Figure 13. The red solid line represents the log-law velocity profile in the inner region ($y/h < 0.2$) where $\bar{u}$ is the point velocity averaged for different experiments, u_* is the shear velocity, y is distance from the top of the sediments, $y_0 = 0.2k_s$ is viscous sublayer, k_s is roughness height which is equal to d_{50}, Br=8.44 is integration constant. The Blue solid line is the velocity profile in the outer region ($y/h > 0.2$) where κ is Von Karman's constant, $\Pi = 0.108$ is the wake strength parameter, and δ is the distance between the bed and the point where maximum velocity occurs, here it is equal to h.

ther accuracy is possible using a more complicated model (then training time increases and needs more datapoints in order to avoid overfitting) but whether it is necessary or not depends on application of the model. The proposed neural network model trained on 4133 experimental dataset applied to the flume experiments in Fig. 10 and Fig. 9, on the numerical simulation in Fig. 11, and field measurements on a river in Illinois state, USA in Fig. 12 and all the comparison, show an accurate prediction of the model, meaning that the model is trained properly and can be used for academic/industrial applications.

5 Conclusion

We applied the U-net with alternating convolutional layers to encode images, followed by alternating convolutional and upsampling layers to decode them and a final convolutional layer with Leaky Relu activation to the velocity field (which was here estimated using a statistical method), and bed topography (measured experimentally). The model contains skip connections, which improves training efficiency. This network has a skipping layer, which allows the decoding part of the network to have additional information about the encoding without losing any information. The final neural network has been established according to the tests performed in order to find out the best parameters, 1400 trainable parameters with no dropout, learning rate of 10^{-3}, and choosing 10^{-6} as the weight decay factor, and using the whole dataset i.e., 4133 data points for the training. The network is summarized in Table. A1.

The trained model does not require solving the shallow water equations numerically. It constructs the solution (bathymetry) by just looking at the input data (velocity fields). Overall, after studying the basic and hyperparameters, and training a neural network model, we have found the best model yields a good accuracy (its less than 1% relative error for estimated bathymetry) when working on the test dataset (laboratory experiments), less than 20% maximum relative error when applying the model to the numerical simulations (using Iber), and with a maximum 26.31% relative error by applying the model to field data (confluence of Kaskaskia River with the Copper Slough).

This work shows the possibility of using the U-net architecture for predicting the bathymetry from the velocity field and provides a user-friendly tool for whoever has the velocity field and is interested in deducing the bed topography. The user of the model does not need to train the model once again, and can download the trained model based on the current work from the provided Github link and apply it to the velocity fields. The user needs to be aware that this is not a general model for every kind of flume or rivers. This work is based on gravel-bed flume/river and therefore the model provides accurate bathymetry for gravel-bed rivers. The strength of the proposed model is that it is trained on experimental data and the neural network model avoids solving the partial differential equations that governs the flow and gives quick access to the bed topography. The training code and best trained model based on the analysis presented here can be found in `Github.com/MehrdadKianiOsh/Deep-learning-bedtopo-vel`.

Appendix A Model architecture

The neural network consists of a series of convolutional blocks and is fully convolutional with 12 layers. A similar structure can be found in all blocks: activation, convolution, batch normalization, and dropout. An upsampling followed by a convolution is used instead of transposing convolutions. Table A1 shows the structure of the U-network used in all of our tests. The network weights were initialized following He et al. (2015). After conducting some tests, we found out that dropout does not aid learning by the model, so we do not use it. Instead, we utilize batch normalization. Training was done using ADAM (Kingma & Ba, 2014) with $\beta_1 = 0.5$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$ parameter values. Learning rate was kept at a constant value during training. A 0.001 learning rate, and a 100 minibatch size were used in all tests.

Appendix B Tsallis entropy-based method

Tsallis introduced generalization of Boltzmann-Gibbs-Shannon (known as Shannon) statistics (Tsallis, 1988). Compared to Shannon entropy-based concepts, the Tsallis entropy-based concepts are either superior or comparable. According to the Tsallis entropy theory, the 2D velocity equation is expressed as follows (Singh, 2016)

$$\frac{u}{u_{max}} = \frac{2}{G}\left[G\frac{y}{h+D}\exp\left(1-\frac{y}{h+D}\right)+\frac{(4-G)^2}{16}\right]^{1/2} - \left(\frac{4-G}{2G}\right) \tag{B1}$$

where the entropic parameter is denoted as G. Here, $D = 0$ (meaning that the maximum velocity happens on the surface flow) because the flume is considered wide $w/h > 3.5$, and on the other hand, Song's experiments emphasizes this hypothesis (Song & Graf, 1996). The definition of G is

$$\frac{u_m}{u_{max}} = \frac{12+G}{24} \tag{B2}$$

The mean velocity in a cross section u_m is known based on reckoning the flow discharge and the section area, while u_{max} occurs at the deepest vertical flow depth and is determined iteratively (by knowing that the maximum velocity happening on the flow surface and assuming the maximum velocity in the first iteration and correcting the maximum velocity iterativaly by calculating the flow discharge and comparing with the groundtruth).

Data Availability Statement

The dataset can be found at `Kaggle.com/MehrdadKianiOsh/bedtopo_vel_fields`, and the python and best model at `Github.com/MehrdadKianiOsh/Deep-learning-bedtopo-vel`.

Table A1. Model summary.

Layer (type)	Output Shape	Function parameters
InputLayer	[1, 256, 64]	
Conv2-D-1	[16, 128, 32]	channel factor=2, convolution 4×4, stride=2, pad=1
BatchNorm2-D-2	[16, 128, 32]	channels×2
Dropout2-D-3	[16, 128, 32]	dropout = 0, inplace=True
LeakyReLU-4	[16, 128, 32]	negative slope=0.2, inplace=True
Conv2-D-5	[16, 64, 16]	channel factor=2, convolution 4×4, stride=2, pad=1
BatchNorm2-D-6	[16, 64, 16]	channels×2
Dropout2-D-7	[16, 64, 16]	dropout = 0, inplace=True
LeakyReLU-8	[16, 64, 16]	negative slope=0.2, inplace=True
Conv2-D-9	[32, 32, 8]	channel factor=4, convolution 4×4, stride=2, pad=1
BatchNorm2-D-10	[32, 32, 8]	channels×4
Dropout2-D-11	[32, 32, 8]	dropout = 0, inplace=True
LeakyReLU-12	[32, 32, 8]	negative slope=0.2, inplace=True
Conv2-D-13	[64, 16, 4]	channel factor=8, convolution 4×4, stride=2, pad=1
BatchNorm2-D-14	[64, 16, 4]	channels×8
Dropout2-D-15	[64, 16, 4]	dropout = 0, inplace=True
LeakyReLU-16	[64, 16, 4]	negative slope=0.2, inplace=True
Conv2-D-17	[64, 8, 2]	channel factor=8, convolution 2×2, stride=2, pad=0
BatchNorm2-D-18	[64, 8, 2]	channels×8
Dropout2-D-19	[64, 8, 2]	dropout = 0, inplace=True
LeakyReLU-20	[64, 8, 2]	negative slope=0.2, inplace=True
Conv2-D-21	[64, 4, 1]	channel factor=8, convolution 2×2, stride=2, pad=0
BatchNorm2-D-22	[64, 4, 1]	channels×8
Dropout2-D-23	[64, 4, 1]	dropout = 0, inplace=True
LeakyReLU-24	[64, 4, 1]	negative slope=0.2, inplace=True
Upsample-25	[64, 8, 2]	scale factor=2, mode='bilinear'
Conv2-D-26	[64, 8, 2]	channel factor=8, convolution 2×2, stride=1, pad=0
BatchNorm2-D-27	[64, 8, 2]	channels×8
Dropout2-D-28	[64, 8, 2]	dropout = 0, inplace=True
ReLU-29	[64, 8, 2]	inplace=True
Upsample-30	[128, 16, 4]	scale factor=2, mode='bilinear'
Conv2-D-31	[64, 16, 4]	channel factor=8, convolution 2×2, stride=1, pad=0
BatchNorm2-D-32	[64, 16, 4]	channels×8
Dropout2-D-33	[64, 16, 4]	dropout = 0, inplace=True
ReLU-34	[64, 16, 4]	inplace=True
Upsample-35	[128, 32, 8]	scale factor=2, mode='bilinear'
Conv2-D-36	[32, 32, 8]	channel factor=4, convolution 3×3, stride=1, pad=1
BatchNorm2-D-37	[32, 32, 8]	channels×4
Dropout2-D-38	[32, 32, 8]	dropout = 0, inplace=True
ReLU-39	[32, 32, 8]	inplace=True
Upsample-40	[64, 64, 16]	scale factor=2, mode='bilinear'
Conv2-D-41	[16, 64, 16]	channel factor=2, convolution 3×3, stride=1, pad=1
BatchNorm2-D-42	[16, 64, 16]	channels×2
Dropout2-D-43	[16, 64, 16]	dropout = 0, inplace=True
ReLU-44	[16, 64, 16]	inplace=True
Upsample-45	[32, 128, 32]	scale factor=2, mode='bilinear'
Conv2-D-46	[16, 128, 32]	channel factor=2, convolution 3×3, stride=1, pad=1
BatchNorm2-D-47	[16, 128, 32]	channels×2
Dropout2-D-48	[16, 128, 32]	dropout = 0, inplace=True
ReLU-49	[16, 128, 32]	inplace=True
Upsample-50	[32, 256, 64]	scale factor=2, mode='bilinear'
Conv2-D-51	[1, 256, 64]	channel factor=1, convolution 3×3, stride=1, pad=1
Dropout2-D-52	[1, 256, 64]	dropout = 0, inplace=True

Acknowledgments

This study was supported by Swiss National Science Foundation with a grant number: 200020_204108 / 1.

References

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.

Bladé, E., Cea, L., Corestein, G., Escolano, E., Puertas, J., Vázquez-Cendón, E., … Coll, A. (2014). Iber: herramienta de simulación numérica del flujo en ríos. *Re-*

vista internacional de métodos numéricos para cálculo y diseño en ingeniería, *30*(1), 1–10.

Bradley, A. A., Kruger, A., Meselhe, E. A., & Muste, M. V. (2002). Flow measurement in streams using video imagery. *Water Resources Research*, *38*(12), 51–1.

Chiu, C.-L. (1987). Entropy and probability concepts in hydraulics. *Journal of Hydraulic Engineering*, *113*(5), 583–599.

Chiu, C.-L. (1988). Entropy and 2-d velocity distribution in open channels. *Journal of Hydraulic Engineering*, *114*(7), 738–756.

Chiu, C.-L. (1989). Velocity distribution in open channel flow. *Journal of Hydraulic Engineering*, *115*(5), 576–594.

Dhont, B. E. M. (2017). *Sediment pulses in a gravel-bed flume with alternate bars* (Tech. Rep.). EPFL.

Durand, M., Andreadis, K. M., Alsdorf, D. E., Lettenmaier, D. P., Moller, D., & Wilson, M. (2008). Estimation of bathymetric depth and slope from data assimilation of swath altimetry into a hydrodynamic model. *Geophysical Research Letters*, *35*(20).

Emery, L., Smith, R., McNeal, D., Hughes, B., Swick, L. W., & MacMahan, J. (2010). Autonomous collection of river parameters using drifting buoys. In *Oceans 2010 mts/ieee seattle* (pp. 1–7).

Ghorbanidehno, H., Lee, J., Farthing, M., Hesser, T., Darve, E. F., & Kitanidis, P. K. (2021). Deep learning technique for fast inference of large-scale riverine bathymetry. *Advances in Water Resources*, *147*, 103715.

Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics* (pp. 249–256).

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press. (`http://www.deeplearningbook.org`)

Griffiths, G. A. (1993). Sediment translation waves in braided gravel-bed rivers. *Journal of Hydraulic Engineering*, *119*(8), 924–937.

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the ieee international conference on computer vision* (pp. 1026–1034).

Honnorat, M., Monnier, J., Rivière, N., Huot, É., & Le Dimet, F.-X. (2010). Identification of equivalent topography in an open channel flow using lagrangian data assimilation. *Computing and visualization in science*, *13*(3), 111–119.

Jaynes, E. T. (1957). Information theory and statistical mechanics. *Physical review*, *106*(4), 620.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Landon, K. C., Wilson, G. W., Özkan-Haller, H. T., & MacMahan, J. H. (2014). Bathymetry estimation using drifter-based velocity measurements on the kootenai river, idaho. *Journal of Atmospheric and Oceanic Technology*, *31*(2), 503–514.

Lewis, Q. W., & Rhoads, B. L. (2015). Resolving two-dimensional flow structure in rivers using large-scale particle image velocimetry: An example from a stream confluence. *Water Resources Research*, *51*(10), 7977–7994.

Lewis, Q. W., & Rhoads, B. L. (2018). Lspiv measurements of two-dimensional flow structure in streams using small unmanned aerial systems: 2. hydrodynamic mapping at river confluences. *Water Resources Research*, *54*(10), 7981–7999.

Liu, X., Song, Y., & Shen, C. (2022). Bathymetry inversion using a deep-learning-based surrogate for shallow water equations solvers. *arXiv preprint arXiv:2203.02821*.

Marcus, W. A. (2002). Mapping of stream microhabitats with high spatial resolution hyperspectral imagery. *Journal of geographical systems*, *4*(1), 113–126.

Moramarco, T., Saltalippi, C., & Singh, V. P. (2004). Estimation of mean velocity in natural channels based on chiu's velocity distribution equation. *Journal of Hydrologic Engineering*, *9*(1), 42–50.

Puleo, J. A., McKenna, T. E., Holland, K. T., & Calantoni, J. (2012). Quantifying riverine surface currents from time sequences of thermal infrared imagery. *Water Resources Research*, *48*(1).

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *International conference on medical image computing and computer-assisted intervention* (pp. 234–241).

Singh, V. P. (2016). *Introduction to tsallis entropy theory in water engineering.* CRC Press.

Singh, V. P., Yang, C. T., & Deng, Z. (2003). Downstream hydraulic geometry relations: 1. theoretical development. *Water resources research*, *39*(12).

Smith, J. D., & McLean, S. (1984). A model for flow in meandering streams. *Water Resources Research*, *20*(9), 1301–1315.

Song, T., Graf, W., & Lemmin, U. (1994). Uniform flow in open channels with movable gravel bed. *Journal of Hydraulic Research*, *32*(6), 861–876.

Song, T., & Graf, W. H. (1996). Velocity and turbulence distribution in unsteady open-channel flows. *Journal of Hydraulic Engineering*, *122*(3), 141-154. doi: 10.1061(asce)0733-9429(1996)122:3(141)

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, *15*(1), 1929–1958.

Tsallis, C. (1988). Possible generalization of boltzmann-gibbs statistics. *Journal of statistical physics*, *52*, 479–487.

Venditti, J., Nelson, P., Minear, J., Wooster, J., & Dietrich, W. (2012). Alternate bar response to sediment supply termination. *Journal of Geophysical Research: Earth Surface*, *117*(F2).

Wilson, G., & Özkan-Haller, H. T. (2012). Ensemble-based data assimilation for estimation of river depths. *Journal of Atmospheric and Oceanic Technology*, *29*(10), 1558–1568.

Wozencraft, J., & Millar, D. (2005). Airborne lidar and integrated technologies for coastal mapping and nautical charting. *Marine Technology Society Journal*, *39*(3).

Yu, S., & Ma, J. (2021). Deep learning for geophysics: Current and future trends. *Reviews of Geophysics*, *59*(3), e2021RG000742.