Supplementary material for reconstruction of three-dimensional turbulent flow structures using surface measurements for free-surface flows based on a convolutional neural network

Anqing Xuan and Lian Shen†

Department of Mechanical Engineering and Saint Anthony Falls Laboratory, University of Minnesota, Minneapolis, Minnesota 55455, USA

1. Effect of number of feature maps

The number of feature maps (or channels) in each CNN layer can affect the expressibility of the CNN. To ensure that our proposed architecture has enough feature maps for flow reconstruction purposes, we consider a model in which, for each intermediate layers other than the input and output, the number of feature maps is increased by 50% compared to the model proposed in the main paper. This new model is trained for the case of $Fr_\tau = 0.08$. Its loss function evaluated over the test set is found to be 0.641, which is comparable to the performance of the original model in the main paper with a test loss of 0.639. This result indicates that the numbers of feature maps used in the original model are adequate for expressing the mappings between the surface and subsurface flow features.

The dimensions of the bottleneck layer in the CNN model, i.e. the output of the encoder and the input of the decoder, are particularly important for reconstructions because it roughly determines how much information is retained by the dimensionality reduction of the encoder. This layer can be considered as a latent representation of the surface features important for subsurface flow reconstructions. If the dimensions of this layer are too small, the discarded information can negatively impact the reconstruction accuracy. Large dimensions may cause unimportant surface features to leak into the latent layer and obfuscate our analyses of the CNN model. The increased dimensions also make the network less efficient. Therefore, the reconstruction performance as a function of the number of channels in the bottleneck layer is studied. As shown in figure 1, the reconstruction errors decrease as the number of feature maps increases. However, the improvement becomes negligible when there are more than 24 layers. Therefore, the present model uses 24 layers at the bottleneck layer, which should retain most of the important features for reconstructions.

Compared to the original input dimensions, $256\times128\times4$, the dimensions of the bottleneck layer, $32 \times 16 \times 24$, are significantly smaller, which suggests that a substantial amount of surface information is in a lower dimensional space. However, we shall note that because our focus is not to investigate the dimensional reduction of the free-surface flows, the output of the encoder still keeps the form of two-dimensional feature maps, instead of reduction to a one-dimensional vector. In other words, there can be spatial correlations in the feature maps that can further reduce the dimensionality of the bottleneck layer.

2. Effect of network depth

In this section, some variations of the network architecture are considered to investigate the effect of the network depth on reconstructions. The configurations of the alternative models

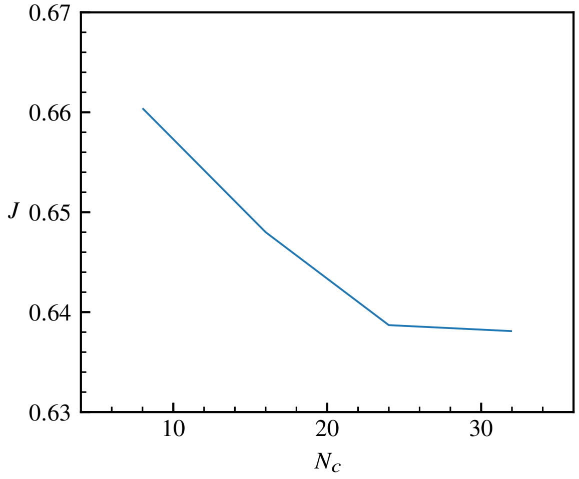

Figure 1: Variations of reconstruction loss J with the number of feature maps in the bottleneck layer (encoder output).

are presented in tables 1–3. Model A adds two more convolution layers in the encoder part of the original CNN model proposed in the main paper; model B adds one convolution layer near the end of the decoder in the original model; model C further adds one convolution layer at the beginning of the decoder compared to model B. Compared to the original model, model A adds approximately 21% more parameters in the encoder; models B and C add 4% and 16% more parameters to the decoder, respectively. These alternative models are trained for the case of $Fr_\tau = 0.08$. The loss function values are computed as a measure of their performances. The losses of model A, B and C are 0.638, 0.646 and 0.645, respectively, which are comparable to the original model with a loss of 0.639. Therefore, we conclude that the original model gains little benefit from increased network depth.

3. Effect of spatial resolutions of surface input and reconstruction output

In the main paper, the input and output of the reconstructions use grids with resolutions lower than the simulation grid. Considering that small-scale flow structures may be filtered out or under-resolved, we consider two CNN models with increased input and output resolutions to investigate whether lower resolutions negatively impact the reconstructions. The first CNN model takes the surface inputs with the original simulation resolution, i.e. on a grid of 256×256, which is higher than the one used in the original CNN model, 256×128; the reconstructions are still performed with the same grid size as in the original model, i.e. $128 \times 64 \times 96$. The second model uses the same input grid size as the original model and outputs the reconstruction on a high-resolution uniform grid of size $256 \times 128 \times 192$, which doubles the number of grid points in each direction compared to the grid used in the original model. As shown in figure 2, the reconstruction accuracy of the original model and that of the two models with higher resolutions are comparable, indicating that the resolution used in the main paper is adequate for reconstruction purposes.

Next, we investigate how lowering the input resolution, which effectively removes small-scale features from the surface inputs, affects the reconstruction results. We consider two coarse input grids, which are twice and fourth coarser than the original input grid used in the main paper, i.e. with grid sizes of 128×64 and 64×32, respectively. The reconstruction errors are compared in figure 3. We can see that the reduced input resolution results in

Input size	Operator	Kernel size	Stride
	Encoder		
$256 \times 128 \times 4$	Convolution	$(3, 3)$	$(1, 1)$
$256 \times 128 \times 4$	Blur pooling	$(3, 3)$	$(2, 2)$
$128 \times 64 \times 16$	Residual block	-	-
$128 \times 64 \times 16$	Convolution	$(3, 3)$	$(1, 1)$
$128 \times 64 \times 16$	Convolution	$(3, 3)$	$(1, 1)$
$128 \times 64 \times 16$	Blur pooling	$(3, 3)$	$(2, 2)$
$64 \times 32 \times 24$	Residual block	-	-
$64 \times 32 \times 24$	Convolution	$(3, 3)$	$(2, 2)$
$128 \times 64 \times 24$	Convolution	$(3, 3)$	$(1, 1)$
$32 \times 16 \times 24$	Residual block	-	-
	Decoder		
$32 \times 16 \times 1 \times 24$	Transposed convolution	$(4, 4, 1)$	$(2, 2, 1)$
$64 \times 32 \times 1 \times 48$	Transposed convolution	$(3, 3, 3)$	$(1, 1, 3)$
$64 \times 32 \times 3 \times 56$	Transposed convolution	$(4, 4, 4)$	$(1, 1, 2)$
$64 \times 32 \times 6 \times 32$	Transposed convolution	$(4, 4, 4)$	$(1, 1, 2)$
$64 \times 32 \times 12 \times 18$	Transposed convolution	$(4, 4, 4)$	$(1, 2, 1)$
$64 \times 64 \times 12 \times 16$	Transposed convolution	$(4, 4, 4)$	$(1, 1, 2)$
$64 \times 64 \times 24 \times 16$	Transposed convolution	$(4, 4, 4)$	$(2, 1, 2)$
$128 \times 64 \times 48 \times 16$	Transposed convolution	$(4, 4, 4)$	$(1, 1, 2)$
$128 \times 64 \times 97 \times 9$	Convolution	$(5, 5, 5)$	$(1, 1, 1)$
$128 \times 64 \times 97 \times 3$	Convolution	$(5, 5, 5)$	$(1, 1, 1)$
$128 \times 64 \times 97 \times 3$	Convolution	$(4, 4, 4)$	$(1, 1, 1)$
$128 \times 64 \times 97 \times 3$	Convolution	$(3, 3, 3)$	$(1, 1, 1)$
$128 \times 64 \times 97 \times 3$	Convolution	$(1, 1, 2)$	$(1, 1, 1)$

Table 1: Alternative CNN architecture A with its parameters, including the input size, kernel size and stride of each block (if applicable). The differences from the original architecture (table 1 in the main paper) are underlined. The input size is written as $N_x \times N_y \times N_c$ for two-dimensional data for the encoder or $N_x \times N_y \times N_z \times N_c$ for three-dimensional data for the decoder, where N_x, N_y and N_z denote the grid dimensions in the x-, y- and z-directions, respectively, and N_c denotes the number of channels. Note that the output of each block is the input of the next block. The output size of the last block is $128 \times 64 \times 96 \times 3$.

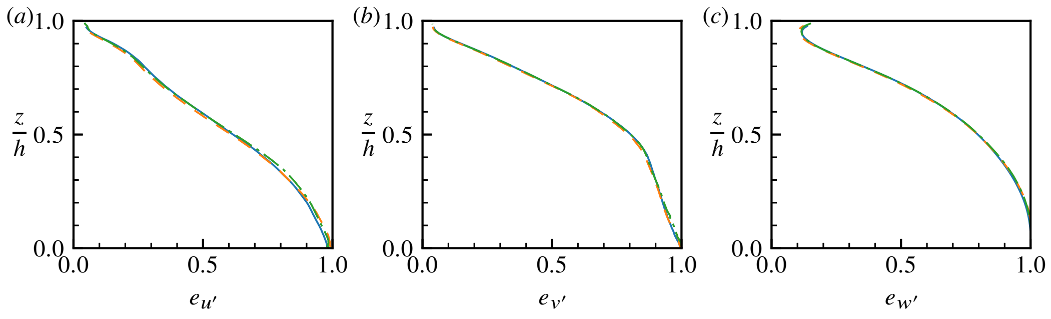

Figure 2: Normalised mean squared reconstruction errors of the CNN model in the main paper with an input grid 256×128 and an output grid $128 \times 64 \times 96$ (——), the CNN model with a dense input grid 256×256 and the same output grid as the original CNN model (– – –) and the CNN model with the same input grid and a dense output grid of size $256 \times 128 \times 192$ (— · —) for the (*a*) streamwise, (*b*) spanwise and (*c*) vertical velocity fluctuations. The case with $Fr_\tau = 0.08$ is plotted.

Input size	Operator	Kernel size	Stride
	Encoder		
$256 \times 128 \times 4$	Convolution	$(3, 3)$	$(1, 1)$
$256 \times 128 \times 4$	Blur pooling	$(3, 3)$	$(2, 2)$
$128 \times 64 \times 16$	Residual block	-	-
$128 \times 64 \times 16$	Convolution	$(3, 3)$	$(1, 1)$
$128 \times 64 \times 16$	Blur pooling	$(3, 3)$	$(2, 2)$
$64 \times 32 \times 24$	Residual block	-	-
$64 \times 32 \times 24$	Convolution	$(3, 3)$	$(2, 2)$
$32 \times 16 \times 24$	Residual block	-	-
	Decoder		
$32 \times 16 \times 1 \times 24$	Transposed convolution	$(4, 4, 1)$	$(2, 2, 1)$
$64 \times 32 \times 1 \times 48$	Transposed convolution	$(3, 3, 3)$	$(1, 1, 3)$
$64 \times 32 \times 3 \times 56$	Transposed convolution	$(4, 4, 4)$	$(1, 1, 2)$
$64 \times 32 \times 6 \times 32$	Transposed convolution	$(4, 4, 4)$	$(1, 1, 2)$
$64 \times 32 \times 12 \times 18$	Transposed convolution	$(4, 4, 4)$	$(1, 2, 1)$
$64 \times 64 \times 12 \times 16$	Transposed convolution	$(4, 4, 4)$	$(1, 1, 2)$
$64 \times 64 \times 24 \times 16$	Transposed convolution	$(4, 4, 4)$	$(2, 1, 2)$
$128 \times 64 \times 48 \times 16$	Transposed convolution	$(4, 4, 4)$	$(1, 1, 2)$
$\underline{128 \times 64 \times 97}$	$\underline{\text{Convolution}}$	$\underline{(5, 5)}$	$\underline{(1, 1)}$
$128 \times 64 \times 97 \times 9$	Convolution	$(5, 5, 5)$	$(1, 1, 1)$
$128 \times 64 \times 97 \times 3$	Convolution	$(5, 5, 5)$	$(1, 1, 1)$
$128 \times 64 \times 97 \times 3$	Convolution	$(4, 4, 4)$	$(1, 1, 1)$
$128 \times 64 \times 97 \times 3$	Convolution	$(3, 3, 3)$	$(1, 1, 1)$
$128 \times 64 \times 97 \times 3$	Convolution	$(1, 1, 2)$	$(1, 1, 1)$

Table 2: Alternative CNN architecture B with its parameters, including the input size, kernel size and stride of each block (if applicable). The differences from the original architecture (table 1 in the main paper) are underlined. The input size is written as $N_x \times N_y \times N_c$ for two-dimensional data for the encoder or $N_x \times N_y \times N_z \times N_c$ for three-dimensional data for the decoder, where N_x, N_y and N_z denote the grid dimensions in the x-, y- and z-directions, respectively, and N_c denotes the number of channels. Note that the output of each block is the input of the next block. The output size of the last block is $128 \times 64 \times 96 \times 3$.

a noticeable increase of error in w' near the surface. The reconstruction errors of u' and v' are increased less significantly than the error of w'. This result indicates that missing information of the small-scale free-surface motions mainly affects the reconstruction of the vertical velocity fluctuations near the surface, which have more small-scale structures than the other two velocity components.

In case of the available data having lower resolutions than the input resolution of the CNN model, one may use interpolations to resample the input, which is commonly used in image processing. To understand how the interpolated inputs affect the reconstruction, we interpolate surface inputs on a grid of size 64×32 onto a grid of size 256×128 with bicubic interpolations and use the interpolated inputs for reconstructions. The reconstruction results are compared with the above CNN model that is trained with the input grid of size 64×32. As shown in figure 4, the reconstructions using interpolated inputs are less accurate than the reconstructions using the model whose native input resolution is 64×32. We note that the interpolation algorithm, e.g. a bilinear interpolation, has a negligible effect on the reconstruction performance. This results indicates that adding the missing small-scale surface features with interpolations can negatively impact the reconstructions and thus should be used with care.

Input size	Operator	Kernel size	Stride
	Encoder		
$256 \times 128 \times 4$	Convolution	$(3, 3)$	$(1, 1)$
$256 \times 128 \times 4$	Blur pooling	$(3, 3)$	$(2, 2)$
$128 \times 64 \times 16$	Residual block	-	-
$128 \times 64 \times 16$	Convolution	$(3, 3)$	$(1, 1)$
$128 \times 64 \times 16$	Blur pooling	$(3, 3)$	$(2, 2)$
$64 \times 32 \times 24$	Residual block	-	-
$64 \times 32 \times 24$	Convolution	$(3, 3)$	$(2, 2)$
$32 \times 16 \times 24$	Residual block	-	-
	Decoder		
$\underline{32 \times 16 \times 1 \times 24}$	$\underline{\text{Convolution}}$	$\underline{(3, 3, 1)}$	$\underline{(1, 1, 1)}$
$32 \times 16 \times 1 \times 48$	Transposed convolution	$(4, 4, 1)$	$(2, 2, 1)$
$64 \times 32 \times 1 \times 48$	Transposed convolution	$(3, 3, 3)$	$(1, 1, 3)$
$64 \times 32 \times 3 \times 56$	Transposed convolution	$(4, 4, 4)$	$(1, 1, 2)$
$64 \times 32 \times 6 \times 32$	Transposed convolution	$(4, 4, 4)$	$(1, 1, 2)$
$64 \times 32 \times 12 \times 18$	Transposed convolution	$(4, 4, 4)$	$(1, 2, 1)$
$64 \times 64 \times 12 \times 16$	Transposed convolution	$(4, 4, 4)$	$(1, 1, 2)$
$64 \times 64 \times 24 \times 16$	Transposed convolution	$(4, 4, 4)$	$(2, 1, 2)$
$128 \times 64 \times 48 \times 16$	Transposed convolution	$(4, 4, 4)$	$(1, 1, 2)$
$\underline{128 \times 64 \times 97}$	$\underline{\text{Convolution}}$	$\underline{(5, 5)}$	$\underline{(1, 1)}$
$128 \times 64 \times 97 \times 9$	Convolution	$(5, 5, 5)$	$(1, 1, 1)$
$128 \times 64 \times 97 \times 3$	Convolution	$(5, 5, 5)$	$(1, 1, 1)$
$128 \times 64 \times 97 \times 3$	Convolution	$(4, 4, 4)$	$(1, 1, 1)$
$128 \times 64 \times 97 \times 3$	Convolution	$(3, 3, 3)$	$(1, 1, 1)$
$128 \times 64 \times 97 \times 3$	Convolution	$(1, 1, 2)$	$(1, 1, 1)$

Table 3: Alternative CNN architecture C with its parameters, including the input size, kernel size and stride of each block (if applicable). The differences from the original architecture (table 1 in the main paper) are $\underline{\text{underlined}}$. The input size is written as $N_x \times N_y \times N_c$ for two-dimensional data for the encoder or $N_x \times N_y \times N_z \times N_c$ for three-dimensional data for the decoder, where N_x, N_y and N_z denote the grid dimensions in the x-, y- and z-directions, respectively, and N_c denotes the number of channels. Note that the output of each block is the input of the next block. The output size of the last block is $128 \times 64 \times 96 \times 3$.

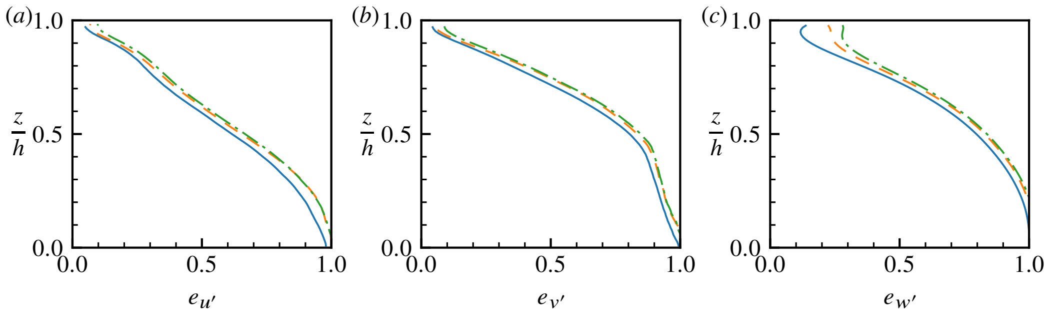

Figure 3: Normalised mean squared reconstruction errors of the CNN models with different input dimensions: the original input grid 256×128 (——), an input grid downsampled by a factor of 2, 128×64 (– – –) and an input grid downsampled by a factor of 4, 64×32 (— · —). The reconstruction errors of the (*a*) streamwise, (*b*) spanwise and (*c*) vertical velocity fluctuations for the case with $Fr_\tau = 0.08$ are plotted.

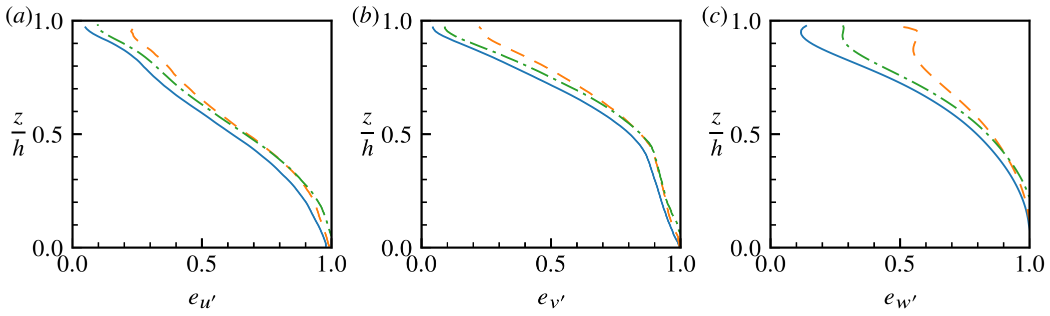

Figure 4: Normalised mean squared reconstruction errors of the reconstructions from the original input of size 256×128 (——) and an input interpolated from 64×32 to 256×128 (– – –). The reconstruction error of a CNN model trained with an input grid of size 64×32 is also compared (— · —). The reconstruction errors of the (*a*) streamwise, (*b*) spanwise and (*c*) vertical velocity fluctuations for the case with $Fr_\tau = 0.08$ are plotted.

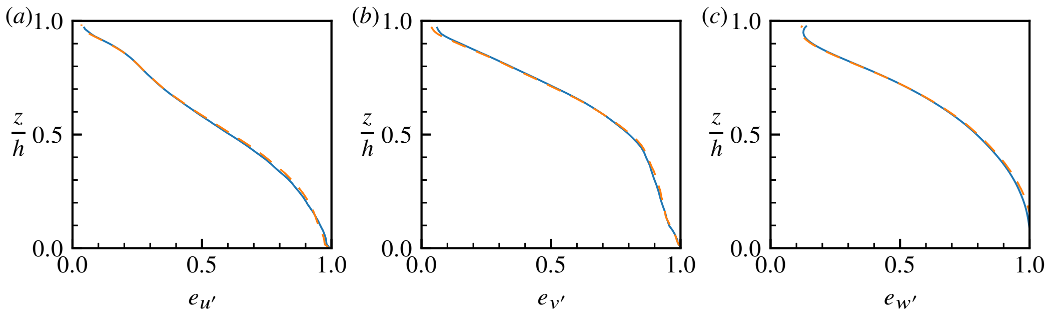

Figure 5: Normalised mean squared reconstruction errors of the CNN models with (——) or without (– – –) the incompressibility constraint. The reconstruction errors of the (*a*) streamwise, (*b*) spanwise and (*c*) vertical velocity fluctuations for the case with $Fr_\tau = 0.08$ are plotted.

4. Reconstructions with and without the incompressibility constraint

In this section, we compare the reconstruction performance of the CNN models with or without the incompressibility constraint. In the main paper, to impose the incompressibility constraint, we add a layer to calculate the curl of a vector field, which essentially let the CNN to predict the vector potential of the velocity field. Here, we consider a model that predicts the velocity field directly. The reconstruction errors plotted in figure 5 indicate that the performance differences between the two methods are negligible. This indicates that predicting the velocity field directly and predicting the vector potential of the velocity field produce equivalent reconstructions.

5. Learning curves

Figure 6 plots the learning curves, i.e. the history of the loss function value during the training of the neural network, for the case with $Fr_\tau = 0.08$. The curves are plotted against training steps. As the training is conducted with a batch size of 8 and the training set has 12, 183 snapshots (see table 3 in the main paper), each epoch has 1523 steps. In the first two epochs, the losses decrease rapidly. Then, the training loss continues decaying while the validation loss plateaus after approximately three epochs and slowly increases afterwards, which indicates that the model starts to overfit, i.e. the model memorizes the features present in the training set but not in the validation set. Physically, this means that the model is

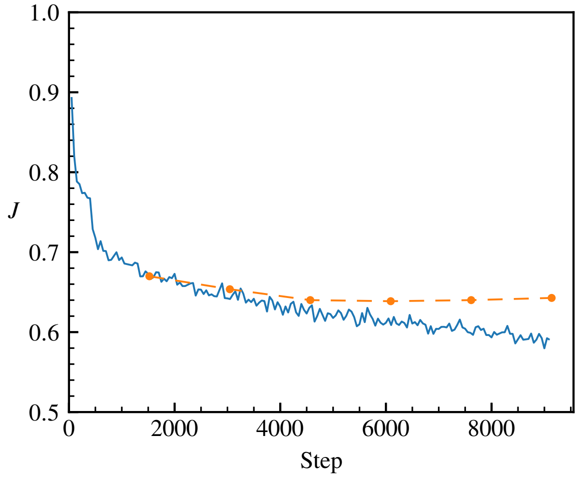

Figure 6: Learning curves showing the training loss (——) and validation loss (– – –) of the case of $Fr_\tau = 0.08$. Note that the validation loss is calculated at the end of an epoch, marked by ●.

learning non-common surface–subsurface relations which apply to the time instants in the training set but do not apply to the instants outside the training set. The training is stopped after the overfitting occurs.