A CNN-supported Lagrangian ISPH model for free surface flow

Ningbo Zhang[a], Shiqiang Yan[a], Qingwei Ma[a*], Xiaohu Guo[b], Zhihua Xie[c]

a School of Science and Technology, City, University of London, London, UK

b Hartree Centre, Science and Technology Facilities Council, Daresbury Laboratory, Warrington, UK

c School of Engineering, Cardiff University, Cardiff, UK

* Author of Correspondence. Email: q.ma@city.ac.uk

Abstract

As a popular method for modelling violent free surface flow, the incompressible Smoothed Particle Hydrodynamics (ISPH) based on the Lagrangian formulation has attracted a great attention worldwide. The Lagrangian ISPH solves the unsteady Navier-Stokes and continuity equations using the projection method, in which the pressure is obtained by solving the pressure Poisson's equation (PPE) that is the most time-consuming part in the ISPH procedure. In this paper, the Convolutional Neural Network (CNN) is combined with ISPH and used to predict the fluid pressure instead of solving the PPE directly. Although limited attempts of using CNN for solving the PPE in Eulerian formulation (referred to as the Eulerian CNN framework) in mesh-based methods are found in the public domain, the present model is the first ISPH model supported by CNN in a Lagrangian formulation. The proposed model overcome several challenges associated with combining CNN with ISPH, including selecting the input parameters, formulating the objective functions, producing the training dataset and dealing with boundary conditions. Two classic free surface problems, i.e. the dam breaking and the wave propagation, are simulated to evaluate the performance of the present model. Quantitative assessments of the numerical error in terms of both the free surface profile and the pressure field are carried out. The assessments show that the new model does not only give results with satisfactory accuracy, but also requires much less computation time for estimating pressure if the number of particles is large, e.g., 100 thousands particles that is usually required in the practical ISPH simulation for free surface flow.

Keywords: Lagrangian ISPH; PPE; machine learning; CNN; free surface flow

1 Introduction

The incompressible Smoothed Particle Hydrodynamics (ISPH) has become a popular method for modelling violent free surface flows (e.g., Shao and Lo, 2003; Lind et al., 2012; Zheng et al., 2014; Gotoh and Khayyer, 2014; Zhang et al., 2018; Khayyer et al., 2021; Luo et al., 2021; Zhang et al., 2021, 2022). The ISPH uses particles, which carry field variables (e.g., the pressure, density and velocity), to discretize the computational domain and solves the incompressible continuity and Navier-Stokes equations using the projection approach, in which a pressure Poisson equation (PPE) is derived and solved to find the fluid pressure. Although the ISPH models were developed using

both Lagrangian particles that move with the material velocity and Eulerian particles that are fixed, modelling the violent free surface flows requires particles to move to conform to the free surface and thus the Lagrangian ISPH (e.g. Zhang et al., 2021,2022) or the hybrid Eulerian-Lagrangian ISPH (e.g. Fourtakas et al., 2018a) are commonly used. Compared with the weakly compressible SPH, where the pressure is estimated using the equation of state, the ISPH generally results in a smoother pressure field, relatively higher convergent rate, larger time step, better energy and volume conservation, as reviewed by Zhang et al. (2022). However, solving the PPE in the ISPH often dominates the computational time.

So far, researchers have been mainly working in two directions to accelerate the ISPH computing. The first one is to develop better numerical schemes, e.g. the Laplacian and gradient discretization schemes, to improve the robustness of solving the PPE and thus the ISPH models. Typical examples include the high-order consistent scheme proposed by Shimizu et al., (2022), the consistent quadric ISPH model developed by Zhang et al. (2022) and the pseudo-spectra ISPH solver proposed by Fourtakas et al. (2021). The applications of these schemes result in a better convergence, compared with the classic ISPH model, e.g. Zheng et al. (2014) and Zhang et al., (2018), and consequently require smaller number of particles and shorter CPU time to achieve required computational accuracy. The second one is to develop massively parallel schemes supporting distributed memory and running in high-performance computer clusters (HPC), such as the message passing interface (MPI) parallelization (Guo et al., 2018; Monteleone et al., 2022) or graphics processing unit (GPU) computing (Chow et al., 2018). These developments allow practical cases with millions of particles to be simulated using HPCs to achieve satisfactory results in a reasonable time frame. It is worth noting that hybrid models, which couple Lagrangian ISPH models with other numerical models such as the fully nonlinear potential (FNPT) model (Yan and Ma, 2017; Fourtakas et al., 2018b) using the domain decomposition coupling strategy, have also demonstrated their effectiveness to improve the overall efficiency.

Recently, the data-driven approach using the machine learning (ML) techniques has been increasingly applied to the fluid simulation (Kutz, 2017) to reconstruct the fluid field from data, e.g., learning the velocity and pressure field from visualization data (Raissi et al., 2020). The ML technique has also been used to replace the challenging and/or time-consuming part of the conventional computational fluid dynamics (CFD) models, such as the turbulence modelling, to save the overall computational time. For the machine-learning based turbulence modelling, an ML algorithm, e.g. the deep neural network, is trained to predict the Reynolds stress anisotropy tensor (Ling et al., 2016) and turbulent production term (Zhang and Duraisamy, 2015) using the database formed by high-fidelity numerical simulation, e.g. direct numerical simulation (DNS) or the large eddy simulation (LES). The trained algorithms are then fed into the Reynolds Averaged Navier-Stokes (RANS) modelling to replace the traditional turbulence models. Compared with the DNS or

LES, the machine-learning supported RANS simulation can largely reserve the turbulence feature but reduce the computational cost significantly (Kutz, 2017). This successful research inspired the present work that aims to develop a Lagrangian ISPH model accelerated by the ML algorithm for predicting the fluid pressure instead of solving PPE.

In literatures, the artificial neural network has been trained to predict the fluid pressure to replace the time-consuming procedure of solving the PPE in the projection-based Navier-Stokes (Yang et al., 2016; Xiao et al., 2018) or Euler equations (Tompson et al., 2017; Dong et al., 2019) in mesh-based models adopting fixed Eulerian grids. Although these ML algorithms may not be directly applied to the Lagrangian ISPH model, reviews on key developments are given herein. Yang et al. (2016) trained the artificial neural network to predict the ground truth pressure with the inputs of the pressure at the previous frame, velocity divergence and the boundary conditions at the current time step. Tompson et al. (2017) introduced the Convolutional Neural Network (CNN) to handle the pressure projection step, aiming to release the dependence on the true solution at the previous frame in the approach developed by Yang et al. (2016), and to minimise the objective using the standard deep-learning optimization approach. Both Yang et al. (2016) and Tompson et al. (2017) proposed to replace the entire procedure of solving the PPE by the trained ML prediction algorithms. Xiao *et al.* (2018) trained a CNN-based deep learning model for solving the large-scale algebraic equations resulting from the discretized PPE, providing the discretization structure and the intermediate velocity field as the input.

In this paper, the CNN model built in the open-access library of Torch7 (Collobert et al., 2011) is used as a ML model. After being trained, the CNN model is then fed to the ISPH model (Zhang et al., 2018) to replace the procedure of solving the PPE, yielding a CNN-supported Lagrangian ISPH, referred to as ISPH_ML in the rest of the paper. As indicated above, the numerical frameworks in Yang et al. (2016) and Tompson et al. (2017) were developed for the Eulerian projection-based methods and did not consider the typical features of the present ISPH modelling for free surface flow, including the Lagrangian nature of the particle movements, the stability of the ISPH (e.g. small fraction of the density variation term introduced in the PPE in Zhang et al., 2018; 2021; 2022) and the free surface boundary conditions. In this work, these features will be considered during training and implementing processes of the CNN model. The ISPH_ML is then evaluated by two benchmark cases, i.e. the dam breaking and the wave propagation. Its convergent behaviour and computational efficiency of the ISPH_ML are also investigated.

It is pointed out that there is limit work on ML supported Lagrangian meshless methods in recent literature. Ladicky et al. (2015) developed an approach to accelerate the SPH using the regression forests. They trained a regressor to update the particle states providing the information of the individual forces acting on the particle. Wessels et al. (2020) developed a Neural Particle Method

for the solution of incompressible free surface flow governed by the Euler equations, in which velocity and pressure are approximated by a neural network. Bai et al. (2021) proposed a Data-Driven SPH (DDSPH) method to implement hydrodynamic modelling in the weakly compressible SPH framework. Marinho (2021) presented a machine learning approach to find the optimal anisotropic SPH kernel, whose compact support consists of an ellipsoid that matches with the convex hull of the self-regulating k-nearest neighbors of a smoothing particle. Also based on the weakly compressible SPH framework, Woodward et al. (2021) presented a learnable hierarchy of parameterized and "physics-explainable" SPH informed turbulent flows simulators using both physics-based parameters and neural networks as universal function approximators. On this basis of Woodward et al. (2021), Chertkov et al. (2022) applied physics-informed machine learning to develop Lagrangian Large Eddy Simulation (L-LES) models described by equations generalizing the weakly compressible SPH formulation with extended parametric and functional freedom for turbulent flows. Based on the Neural Particle Method (NPM), Bai et al. (2022) developed a general Neural Particle Method (gNPM) for viscous hydrodynamics modeling. Li and Farimani (2022) applied the graph neural network to accelerate the Lagrangian fluid simulation. To the best of our knowledge, the ISPH_ML is the first CNN-supported Lagrangian ISPH method.

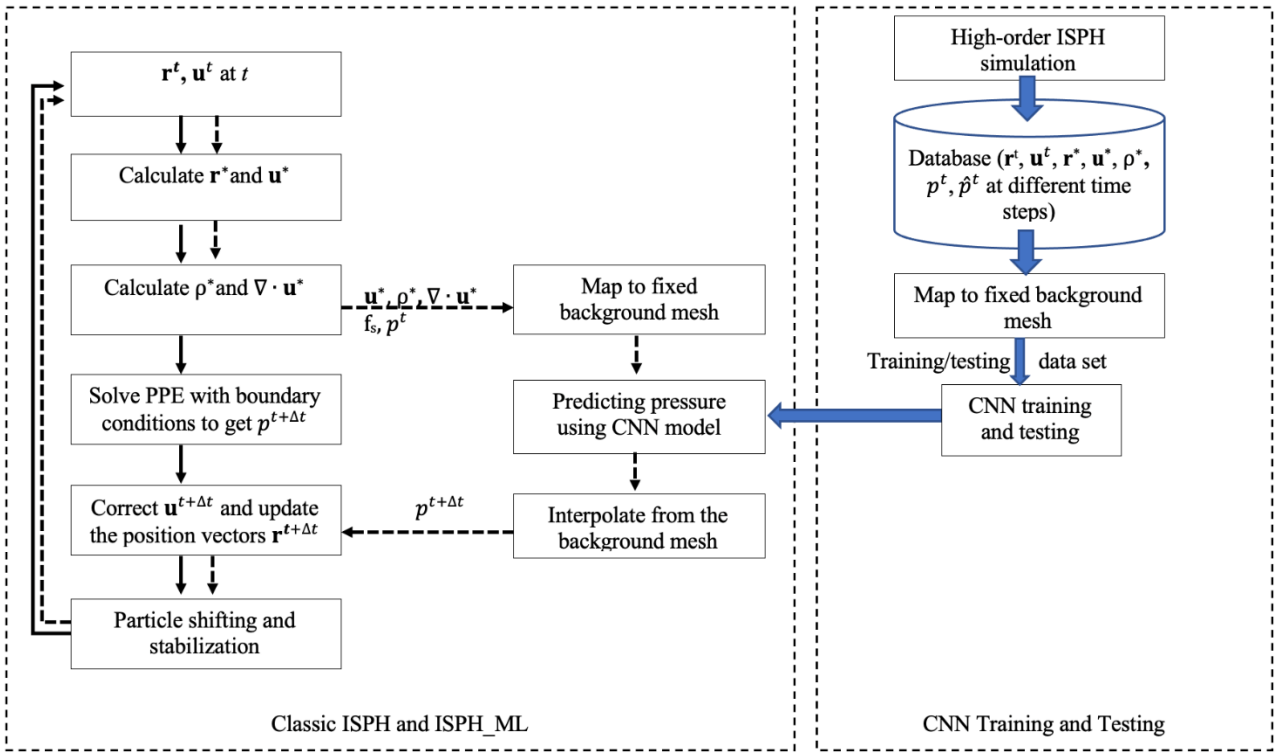


Fig. 1 Framework of CNN-supported Lagrangian ISPH (left: flowcharts of the classic Lagrangian ISPH (solid arrows) and the ISPH_ML (dashed arrows); right: CNN training and testing procedure)

2. Numerical Framework for CNN-Supported Lagrangian ISPH

As indicated above, the PPE for solving the pressure is replaced by an CNN based ML algorithm in

ISPH_ML proposed in this paper. The framework of ISPH_ML is illustrated in Fig. 1, which includes the flowcharts for solving the fluid flow and the procedure for training and testing ML algorithm. Each of the elements in the framework will be discussed below.

2.1. Lagrangian ISPH

The formulation of ISPH is well known and has been described in many references cited above. Only a brief discussion on it will be given here for completeness. Readers are referred to other references, e.g. Khayyer et al. (2020), Luo et al. (2021), Zhang et al. (2022) and so on, for more details. The ISPH solves the incompressible NS equation and the continuity equation,

$$\frac{D\mathbf{u}}{Dt} = -\frac{1}{\rho}\nabla p + \mathbf{g} + \nu\nabla^2\mathbf{u}\,, \tag{1}$$

$$\nabla\cdot\mathbf{u} = 0\,, \tag{2}$$

where D/Dt is the material derivative following fluid particles and ∇ is the gradient operator; ρ is the fluid density; $\mathbf{u}$ is the particle velocity; p is the pressure; $\boldsymbol{g}$ is the gravitational acceleration; and ν is the kinematic viscosity. On the solid boundaries, the following boundary conditions are imposed,

$$\mathbf{u}\cdot\mathbf{n} = \mathbf{U}\cdot\mathbf{n}, \tag{3}$$

$$\mathbf{n}\cdot\nabla p = \rho\left(\mathbf{n}\cdot\mathbf{g} - \mathbf{n}\cdot\dot{\mathbf{U}}\right), \tag{4}$$

in which $\mathbf{n}$ is the unit normal vector of the solid boundary; $\mathbf{U}$ and $\dot{\mathbf{U}}$ are the velocity and acceleration of the solid boundary. On the free surface, the following condition is applied

$$p = 0\,. \tag{5}$$

The governing equations and boundary conditions are solved using a projection method. Assuming the position ($\mathbf{r}^t$) and the velocity ($\mathbf{u}^t$) of fluid particles at time t are known, the pressure, velocity and position of the fluid particles at the new time step, $t + \Delta t$, where Δt is the time step size, can be predicted using three stages,

(1) Prediction (intermediate) stage: an intermediate temporal velocity $\mathbf{u}_i^*$ and position $\mathbf{r}_i^*$ of particle i are predicted using

$$\mathbf{u}_i^* = \mathbf{u}_i^t + (\mathbf{g} + \nu\nabla^2\mathbf{u}_i^t)\Delta t\,, \tag{6}$$

$$\mathbf{r}_i^* = \mathbf{r}_i^t + \mathbf{u}_i^*\Delta t, \tag{7}$$

and the intermediate density ρ^* at the particle i is calculated by

$$\rho_i^* = \sum_{j=1}^{N} m_j W\left(\mathbf{r}_{ij}^*\right)\,, \tag{8}$$

where N is the number of neighbouring particles of the particle i, m_j is the particle mass of the local particle j and $W\left(\mathbf{r}_{ij}\right)$ is a kernel function corresponding to the position vector $\mathbf{r}_{ij} =$

$\mathbf{r}_i - \mathbf{r}_j$.

(2) Pressure projection stage: the pressure at the time step $p^{t+\Delta t}$ is governed by

$$\nabla^2 p^{t+\Delta t} = \frac{\rho \nabla \cdot \mathbf{u}^*}{\Delta t}, \tag{9a}$$

which is obtained by inserting Eqs. (6) and (7) into Eqs. (1) and (2).

According to numerical tests presented in Ma and Zhou (2009) and also suggested by Zheng et al. (2014), the better results for violent water waves can be obtained by using a mixed term on the right-hand side of Eq. (9a). Following this, an alternative form (Zhang et al., 2018; 2021; 2022) is applied to all internal fluid particles,

$$\nabla^2 p_i^{t+\Delta t} = \Psi / \Delta t, \tag{9b}$$

where $\Psi = \alpha \frac{\rho - \rho^*}{\Delta t} + (1-\alpha)\rho \nabla \cdot \mathbf{u}^*$ and α is the blending coefficient. When $\alpha = 0$, Eq. (9b) is the same as Eq. (9a). Following Zhang et al. (2018; 2021; 2022), α is taken as 0.01 in this work. On the solid boundary, Eq. (4) is imposed. Different schemes are available to impose the solid boundary condition. In this work, the solid boundaries are represented using the solid particles and the mirror particles generated at each time step following the movement of the particles near the solid boundaries (Zhang et al., 2018). Eq. (5) are imposed to the particles on the free surface, which are identified using the technique developed by Zheng et al. (2014).

(3) Correction stage: the velocity $\mathbf{u}_i^{t+\Delta t}$ and the position vector $\mathbf{r}_i^{t+\Delta t}$ of particle i at $t + \Delta t$ are corrected using

$$\mathbf{u}_i^{t+\Delta t} = \mathbf{u}_i^* - \frac{\mathbf{1}}{\boldsymbol{\rho}} \nabla p_i^{t+\Delta t} \Delta t, \tag{10}$$

$$\mathbf{r}_i^{t+\Delta t} = \mathbf{r}_i^t + \frac{\mathbf{u}_i^t + \mathbf{u}_i^{t+\Delta t}}{\mathbf{2}} \Delta t \quad . \tag{11}$$

The hybrid particle stabilization scheme proposed by Zhang et al. (2018) is also employed in this paper.

2.2. Numerical Implementation in Classic ISPH

Numerical discretisation schemes are needed to implement the procedure. For this purpose, the velocity divergence and the viscous stress term are discretised at the particle i, respectively, by

$$\nabla \cdot \mathbf{u}_i = -\frac{1}{\rho_i} \sum_{j=1}^{N} m_j (\mathbf{u}_i - \mathbf{u}_j) \cdot \nabla_i W(\mathbf{r}_{ij}), \tag{12}$$

$$\nabla \cdot (\nu_i \nabla \mathbf{u}_i) = \sum_{j=1}^{N} 8 m_j \left(\frac{\nu_i + \nu_j}{\rho_i + \rho_j} \frac{\mathbf{u}_{ij} \cdot \mathbf{r}_{ij}}{d_{ij}^{\,2} + \varepsilon^2} \right) \cdot \nabla_i W(\mathbf{r}_{ij}), \tag{13}$$

where ϵ is a small number to avoid the singularity caused by $\mathbf{r}_{ij} = 0$; $\mathbf{u}_{ij} = \mathbf{u}_i - \mathbf{u}_j$. These equations are commonly used in other SPH applications, e.g. Cleary and Monaghan (1999) and

Monaghan (2005). The pressure gradient in the correction stage and the left-hand side of the pressure boundary condition (Eq. (4)) are discretised by using the linear semi-analytical finite difference interpolation scheme (Zheng et al., 2018; Zhang et al., 2021;2022). For 2D problems, it reads

$$\nabla p_i = \sum_{j=1,j\neq i}^{N} \frac{n_i^{x_k} B_{ij}^{x_m} - n_i^{xy} B_{ij}^{x_k}}{n_i^{x} n_i^{y} - n_i^{xy} n_i^{xy}} (p_j - p_i), \tag{14}$$

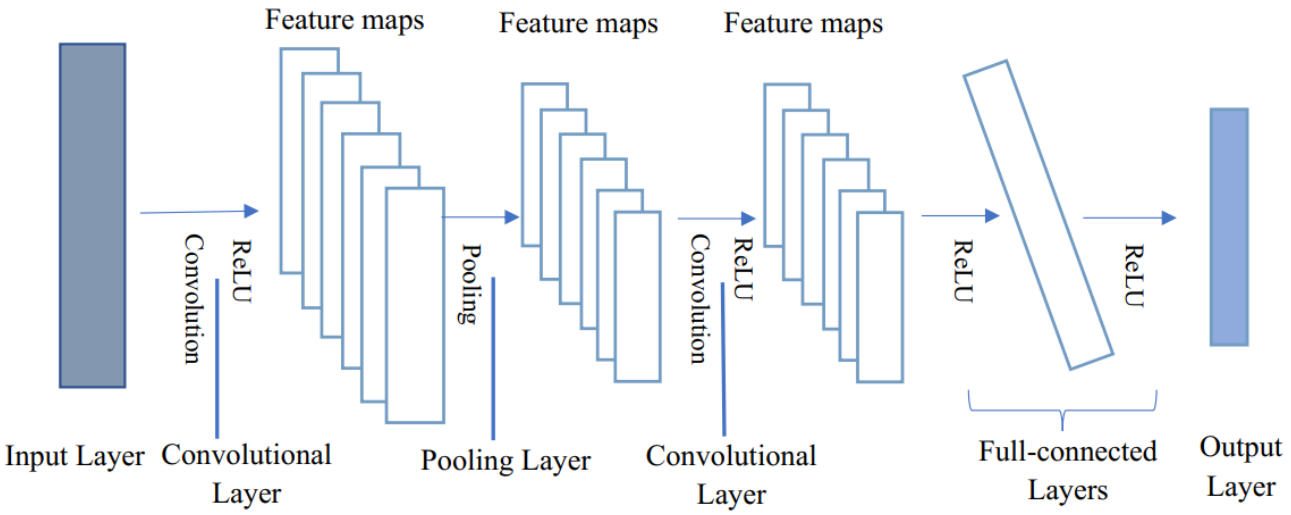
where $n_i^{xy} = \sum_{j=1,j\neq i}^{N} \frac{\left(\mathbf{r}_j^{x_m} - \mathbf{r}_i^{x_m}\right)\left(\mathbf{r}_j^{x_k} - \mathbf{r}_i^{x_k}\right)}{d_{ij}^2} W(\mathbf{r}_{ij})$, $n_i^{x_m} = \sum_{j=1,j\neq i}^{N} \frac{\left(\mathbf{r}_j^{x_m} - \mathbf{r}_i^{x_m}\right)^2}{d_{ij}^2} W(\mathbf{r}_{ij})$ $B_i^{x_m} = \sum_{j=1,j\neq i}^{N} \frac{\left(\mathbf{r}_j^{x_m} - \mathbf{r}_i^{x_m}\right)}{d_{ij}^2} W(\mathbf{r}_{ij})$, $x_m = x$ when $x_k = y$ or $x_m = y$ when $x_k = x$, and $\mathbf{r}^{x_m}$ is the component of the position vector in x_m direction. The Laplacian in Eq. (9b) is discretised as

$$\nabla^2 p_i = \sum_{j=1}^{N} \frac{4m_j}{\rho_i + \rho_j} \frac{(p_i - p_j)\mathbf{r}_{ij}}{d_{ij}^2 + \epsilon^2} \cdot \nabla_i W(\mathbf{r}_{ij}). \tag{15}$$

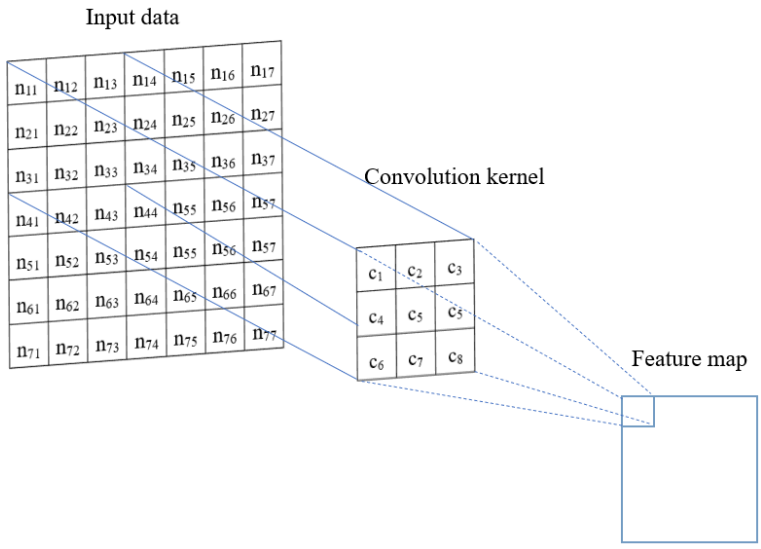
The ISPH implemented using Eq. (9b) together with Eqs. (12) – (15) is referred to as the classic ISPH (Zhang et al., 2018) in the paper for the convenience. The corresponding flowchart is illustrated by the solid arrows in Fig. 1.

2.3 CNN-supported Lagrangian ISPH

As shown in Fig.1, the main difference between the classic and ISPH_ML lies in the way to find the pressure. In the classic ISPH, the pressure is found by solving Eq. (9b), while it is predicted by a ML in ISPH_ML. This section will discuss the issues related to coupling the ML with ISPH, training and testing the ML, and implementing the ML. The model for ML is based on CNN in this paper.



(a) CNN structure



(b) Convolutional operation

Fig. 2 Illustration of convolutional neural network (CNN)

2.3.1. Coupling CNN with Lagrangian ISPH

The CNN, introduced by LeCun et al. (1990), is inspired by the natural visual perception mechanism of the living creatures and has been shown to have extraordinary performance for dealing with many problems. The network is composed of four parts as shown in Fig. 2 (a): (i) input layer; (ii) convolutional layers; (iii) fully-connected layers and (iv) output layer. The convolutional layers extract the main features of the input data. The fully-connected layers build the relations between the main features and output variables as done by a normal neural network.

Each of the convolutional layers usually consists of convolution, activation (or application of nonlinearity) and pooling. Convolution performs the convolutional operation (i.e., dot product of matrixes) between the input data and a set of learnable filters, which also called as kernels. A kernel is a square matrix of discrete numbers, with a typical size of 3 × 3, as illustrated in Fig. 2(b), which is also used in this paper. Dot product of the kernel with corresponding elements in the input data gives one value representing a feature of data. With sliding the kernel over all the input data, a feature map is formed. Performing the similar convolution for each of kernels, a number of the feature maps are obtained as seen in Fig. 2(a). A featured value at location (*i*, *j*) in the *k*th feature map of *l*th layer, $h^l_{i,j,k}$, can be mathematically expressed as

$$h^l_{i,j,k} = \mathbf{w}^{l\,T}_k \mathbf{x}^l_{i,j} + b^l_k \tag{16}$$

where $\mathbf{w}^l_k$ and b^l_k are the weight vector and bias term of the *k*th kernel of the *l*th layer respectively, and $\mathbf{x}^l_{i,j}$ is the input patch centered at location (*i*, *j*) of the *l*th layer. During or after the convolution,

activation (or application of nonlinearity) is usually performed in order to introduce the nonlinearities. Among the possible activation options, the Rectified linear unit (ReLU) function is the most popular and used in this paper, which basically converts all the negative feature values into zero. After convolution with the activation, the pooling may be applied to the feature maps, which basically means selecting representative features and discarding others. Often used method, also employed in this paper, for pooling is to select the maximum in a 2×2 submatrix of a feature map. This part aims to reduce the dimension of the data, speeds up the process and eliminate the issue of overfitting.

After performing one or more convolution layers, a number of features maps are extracted. On this basis, one or more fully-connected layers may be applied to establish the relation between the features (treated as inputs) and the output variables (which is the pressure values in this work), with learnable weights, in a way similar to a normal neural network.

After training using the structure illustrate in Fig. 2(a), a relation (a kind of implicit function) between the input variables and the output is established, in which the relative importance of variables is reflected by weighting in the network coefficients (denoted as c_p) that are determined in the convolutional and fully-connected layers. For convenience, the relation is named as f_{CNN} hereafter. More information about CNN can be referred to Gu et al. (2018) and Pinaya et al. (2020).

In the ISPH_ML, the trained CNN model is used to predict the pressure. The flowchart of the ISPH_ML has been illustrated by the dashed arrows in Fig. 1. Procedure for coupling CNN model with ISPH is summarised below with a slightly more details than the figure:

(1) Find the intermediate velocity $\mathbf{u}_i^*$ and position $\mathbf{r}_i^*$ of particle i using Eq. (6) and (7), respectively, after finding the solution at time t.

(2) Calculate the intermediate density ρ^* and velocity divergence $\nabla \cdot \mathbf{u}^*$ using Eq. (8) and Eq. (12), respectively;

(3) Identify the free surface particles using the method proposed by Zheng et al. (2014) and generate the instantaneous solid particles and corresponding mirror particles using the method developed by Zhang et al. (2018);

(4) Predict the pressure $p_i^{t+\Delta t}$ at fluid particles (excluding all the boundary particles) using the trained CNN model with the pressure at the free surface particle is enforced to be zero, i.e. Eq. (5);

(5) Estimate the pressure $p_b^{t+\Delta t}$ at solid boundary particles with position vector of $\mathbf{r}_b$ using the pressure at the neighbouring fluid particles by (Adami et al., 2012),

$$p_b = \frac{\sum_{j=1}^{N} p_j W(\mathbf{r}_{bj}) - (\mathbf{g} - \dot{\mathbf{U}}) \cdot \sum_{j=1}^{N} \rho_j (\mathbf{r}_j - \mathbf{r}_b) W(\mathbf{r}_{bj})}{\sum_{j=1}^{N} W(\mathbf{r}_{bj})}, \quad (17)$$

in which N is the total number of fluid particles in the influence domain of Particle b.

(6) Correct the velocity $\mathbf{u}_i^{t+\Delta t}$ and the position vector $\mathbf{r}_i^{t+\Delta t}$ of all the particles at $t + \Delta t$ using Eq. (10) and Eq. (11), respectively, where the pressure gradient in Eq. (10) is evaluated using Eq. (14).

Compared with the classic ISPH formulated using Eqs. (1) – (15), the distinguished feature of ISPH_ML is the way to evaluate the pressure $p_i^{t+\Delta t}$ i.e., Steps (4) – (5), using the CNN model to replace directly solving the PPE (Eq. (9b)).

The CNN model is trained with the objective of minimising the difference between the predicted pressure and the corresponding data in the training dataset. However, due to the nature of machine learning algorithm, the pressure predicted by ML may not be exactly same as the solution obtained by directly solving Eq. (9b) (Tompson et al., 2017). There may be many reasons for this to happen. One of them is related to implementation of boundary conditions: i.e. the free surface and solid boundary conditions, that play a critical role on the water wave problems. In the existing work (Yang et al., 2016; Hasegawa et al., 2020; Peng et al., 2021), the boundary conditions are included in the training process and prediction model. Following the practice, the information about free surface particles is involved in both the training and prediction process in our work. To do so, a flag denoted by f_s is created. f_s = 1 for free surface particles and f_s = 0 otherwise. f_s will be fed to the CNN model as a part of input parameters during training and predicting processes. When predicting the pressure at Step (4), the pressure at all free surface particle with f_s = 1 is enforced to be zero. This ensures that the free surface boundary condition, i.e. Eq. (5), is satisfied. As for the solid boundary condition for pressure, we tried similar thing but we found that the larger error would be observed if the Neumann boundary conditions on the solid boundary (Eq. (4)) are imposed during training and testing stage. To avoid this, the solid boundary is not imposed during the training/testing. When implementing the trained model to predict the pressure, the pressure at the particles on the solid boundary are estimated directly using Eq. (17). This does not only ensure a satisfactory implementation of the solid boundary condition (Eq. (4)), but also avoid involving extra input parameter to indicate the solid particles and consequently, improve the efficiency of the ML model. This may not be perfect and need to be improved in future. However, the results presented below will show that the treatment is satisfactory.

The second one is that the overall accuracy of f_{CNN} is affected by the database to be used for training and by the parameters involved in deriving the f_{CNN}. To produce the database with high accuracy, we use the consistent quadric ISPH (ISPH_CQ), developed by the authors of this paper (Zhang et al., 2022) and to be briefed in the following section. The method has quadric convergent property and

results in a significantly higher accuracy than the classic ISPH formulated by Eqs. (1) – (15) shown in the cited paper. This is analogy to the idea adopted in the turbulence modelling (Ling et al., 2016; Zhang and Duraisamy, 2015) that use the higher-fidelity modelling solutions (such as the DNS and the LES results) to build a model used by lower-fidelity RANS simulation. The parameters for deriving f_{CNN} should reflect the hydrodynamic characteristics, which will be discussed in Section 2.3.4.

2.3.2. Training and testing database

As indicated above, we use the ISPH_CQ to produce the numerical results for the training and testing database. The governing equations, boundary conditions and procedure of the ISPH_CQ are the same as the classic ISPH illustrated in Fig. 1, but the numerical discretisation schemes for the gradient and Laplacian operations are formulated by the quadric SFDI (Yan et al., 2020), which are given by

$$\nabla p_i = \sum_{j=1}^{N} \boldsymbol{\Phi}_{ji}^{g}\left(p_j - p_i\right), \tag{18}$$

$$\nabla^2 p_i = \boldsymbol{I}^{\mathrm{T}} \sum_{j=1}^{N} \boldsymbol{\Phi}_{ji}^{s}\left(p_j - p_i\right), \tag{19}$$

respectively, where $\boldsymbol{I} = [1 \quad 1 \quad 1]^{\mathrm{T}}$ and

$$\boldsymbol{\Phi}_{ji}^{g} = \boldsymbol{M}_{1q,i}^{-1}\left(\frac{W(\mathbf{r}_{ji})}{d_{ji}^2}\boldsymbol{q}_{ji} - \sum_{k=1}^{N}\frac{W(\mathbf{r}_{ki})}{d_{ki}^2}\boldsymbol{q}_{ki}\left(\mathbf{r}_{ki}^{(2c)}\right)^{\mathrm{T}}\boldsymbol{M}_{2c,i}^{-1}\frac{W(\mathbf{r}_{ji})}{d_{ji}^4}\mathbf{r}_{ji}^{(2c)} - \sum_{k=1}^{N}\frac{w_{kI}}{d_{ki}^2}\boldsymbol{q}_{ki}\boldsymbol{\Pi}_{ki}^{\mathrm{T}}\boldsymbol{M}_{2s,i}^{-1}\boldsymbol{\Gamma}_{ji}\right) \tag{20}$$

$$\boldsymbol{\Phi}_{ji}^{s} = 2\boldsymbol{M}_{2s,i}^{-1}\left(\boldsymbol{\Gamma}_{ji} - \sum_{k=1}^{N}\frac{w_{kI}}{d_{ki}^4}\boldsymbol{\Pi}_{ki}\boldsymbol{G}_{ki}^{\mathrm{T}}\boldsymbol{\Phi}_{ki}^{g}\right), \tag{21}$$

where $\mathbf{r}_{ji}^{(2s)} = \left[x_{ji}^2 \quad y_{ji}^2 \quad z_{ji}^2\right]^{\mathrm{T}}$, $\mathbf{r}_{ji}^{(2c)} = [x_{ji}y_{ji} \quad x_{ji}z_{ji} \quad y_{ji}z_{ji}]^{\mathrm{T}}$ and d_{ji} is the distance between particle i and its neighbouring particle j. The definitions of matrices in Eq. (19) and (20), including $\boldsymbol{M}_{2c,i}$, $\boldsymbol{\Pi}_{ji}$, $\boldsymbol{M}_{2s,i}$, $\boldsymbol{G}_{ji}$, $\boldsymbol{q}_{ji}$, $\boldsymbol{\Gamma}_{ji}$ and $\boldsymbol{M}_{1q,i}$, can be found in Yan et al. (2020) and Zhang et al. (2021, 2022). The velocity divergence and the viscous term are discretised, respectively, by

$$\nabla \cdot \mathbf{u}_i = \sum_{j=1}^{N} \boldsymbol{\Phi}_{ji}^{g} \cdot \left(\mathbf{u}_j - \mathbf{u}_i\right), \tag{22}$$

$$\nabla \cdot (\nu_i \nabla \mathbf{u}_i) = \sum_{j=1}^{N} \nu_i \boldsymbol{\Phi}_{ji}^{s} \cdot \left(\mathbf{u}_j - \mathbf{u}_i\right). \tag{23}$$

For each free surface problem considered in this paper, a random selection of the test conditions will be considered. For example, in the dam breaking problem, we use different values of the water column hight and width to form a series of cases for the database, which are randomly selected in a practical range. Each case will be run by using the ISPH_CQ and the particle position vector $\mathbf{r}^t$, velocity $\mathbf{u}^t$, intermediate position $\mathbf{r}^*$, intermediate velocity $\mathbf{u}^*$, intermediate density ρ^* and fluid pressure p^t at specific timesteps will be recorded. From these field data, one can post-process other parameters if required, e.g. the velocity divergence and the dynamic pressure field $\hat{p}^t$. The flowchart of training and testing the CNN model is illustrated on the right column of Fig. 1.

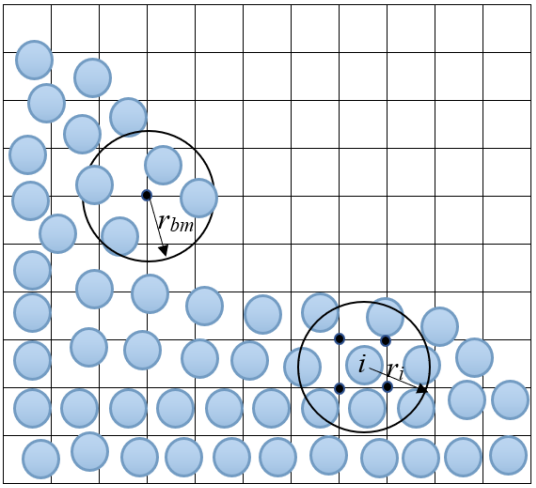


Fig. 3 Illustration of the Background Mesh and ISPH particles

2.3.3 Background Mesh mapping particle-based data with grid-structured CNN architecture

As illustrated in Fig.2, the convolution operation and the neural networks in CNN are specifically designed to work with grid-structured inputs which include strong spatial dependencies in local regions of the grid (Pinaya et al., 2020). However, in the Lagrangian ISPH simulation, particles are moving following their material velocities and may be artificially shifted for achieving better particle stabilisation and satisfactory particle distribution. The above-mentioned field data resulted from the ISPH simulation corresponds to the moving particle positions which are irregularly distributed generally. Such dataset cannot be directly dealt with by the CNN model. To overcome this problem, a background mesh (Fig.3) is introduced. The field data corresponding to irregularly distributed particles will be mapped to that at the node of the background mesh by using an appropriate interpolation. The latter will be used in the CNN training and runtime prediction. The background mesh covers the entire computational domain and the mesh size is taken as the initial particle spacing *dx*. Once the background mesh is established in the initial time step, it will not be changed for all other time steps. One may use high-order interpolation, for the convenience of the ISPH implementation, we use the kernel interpolation. For any node *I* in the background mesh, the physical quantity $fg_{,I}$ is estimated using

$$\mathrm{f}_{g,I} = \sum_{j=1}^{N} f_{p,j} W(\boldsymbol{r}_{Ij}) \,/\, \sum_{j=1}^{N} W(\boldsymbol{r}_{Ij}) \tag{24}$$

where *N* is the number of ISPH particles in the influence domain of Node *I*, which is a circular region centred at *I* with a radius of r_{bm}; $f_{p,j}$ is the corresponding physical quantity at the ISPH particle with a local particle number *j*. After the mapping, the CNN uses the field data corresponding to the background mesh to proceed the learning and training, resulting a trained CNN function f_{CNN}. During the runtime prediction, the trained CNN model is also expected to receive the same grid-structured data. After the pressure at all nodes of the background grid is predicted using

the trained CNN model, the kernel-based interpolation is used again to give the pressure at the ISPH particles,

$$f_{p,i} = \sum_{J=1}^{N} f_{g,J} W(\boldsymbol{r}_{iJ}) / \sum_{J=1}^{N} W(\boldsymbol{r}_{iJ}), \tag{25}$$

where $f_{p,i}$ is the physical quantity, i.e. the predicted pressure, at the particle i, N is the number of background nodes in the influence domain of the particle i with a radius of r_i; $f_{g,J}$ is the corresponding value at the background node with a local number J. It is noted that the interpolation seems to not work well in the region that particles are dispersed due to violent motion of the free surface, such as splashing. However, these dispersed particles will be judged as free surface particles on which pressure will be imposed to be zero, and thus the accuracy of the interpolation does not actually affect the results. The idea of using background mesh to assist and support the particle methods is not new. In the classic ISPH (Zhang et al., 2018) and the ISPH_CQ (Zhang et al., 2022), it has been used to assist the particle searching. It is also used to enhance the δ-SPH for ocean engineering applications (You et al., 2021). However, in the present numerical framework and the ISPH_ML, it is an essential step to make satisfactory data exchange between the ISPH and the CNN.

2.3.4 Input parameters and predicted pressure

Yang et al. (2016) proposed to use the pressure at the previous time step and the intermediate velocity divergence $\nabla \cdot \mathbf{u}^*$ at the current step, together with the solid boundary conditions that are realised by a geometry field, as the input data. Tompson et al. (2017) suggested to release the dependence on the solution at the previous time step, as it may cause accumulation of the error. They only used the intermediate velocity divergence $\nabla \cdot \mathbf{u}^*$ and the geometry field as the input data. Despite the facts that Yang et al. (2016) and Tompson et al. (2017) designed the training for the PPE in the Eulerian mesh-based methods, we select the input parameters based on their work.

The intermediate velocity divergence $\nabla \cdot \mathbf{u}^*$ in Yang et al. (2016) and Tompson et al. (2017) corresponds to the right-hand side of the PPE, i.e. $\frac{\rho \nabla \cdot \mathbf{u}^*}{\Delta t}$ in Eq. (9a). However, the PPE in the present ISPH model is written as a blended form, i.e. Eq. (9b), of which the right hand side is $\Psi/\Delta t$. The introduction of the density variation term is to improve the numerical stability of the ISPH. Analogically, we use Ψ as one of the input parameters. Following Yang et al. (2016), we also consider the pressure at the previous time step as we found in our preliminary tests that involvement of the pressure at the previous time step can improve the prediction of the pressure. The results will be discussed in the following section. Furthermore, the water wave is driven by the gravity and the viscous effects may be significant in the case with breaking waves. These effects are taken into

account through the calculation of the intermediate velocity, $\mathbf{u}^*$, we also involve $\mathbf{u}^*$ as one of input parameters. With this a better performance of the ISPH-ML can be achieved than without considering $\mathbf{u}^*$ as an input parameter. The geometry field is not needed here, which are critical in Yang et al. (2016) and Tompson et al. (2017) to specify the solid boundary, since the pressure boundary condition is approximated using Eq. (17). However, as indicated above, the flag specifying the free surface particle is required to consider the free surface boundary condition.

Unlike the existing works (Yang et al., 2016; Tompson et al., 2017; Xiao et al., 2018; Dong et al., 2019), in the present work, the dynamic pressure p_d instead of the total pressure is learnt during the training phase and predicted in the runtime prediction. The total pressure is then obtained by summing the predicted dynamic pressure and the static pressure. The static pressure can be directly calculated using the particle position without the need of prediction. There are two reasons for doing so. One is that the dynamic pressure is significantly smaller than the total pressure in a large part of the fluid domain. It dominates only near the free surface and during a short duration of violent impact. However, the dynamic pressure is critical for achieving a satisfactory accuracy for the gravity driven free surface flow. Therefore, a model trained toward a minimized error in the dynamic pressure can reduce the adverse effect of the prediction error by the CNN. Secondly, the static pressure is only correlated with the position of the particles and can be evaluated exactly. If it would be predicted by CNN model, the error in estimating this part would be unnecessarily introduced.

Based on the above discussions, the field data of Ψ, $\mathbf{u}^*$, f_s and p_d^t are used as the input data in order to learn a model to predict the dynamic pressure $\hat{p}_d^{t+\Delta t}$. The function f_{CNN} from input to output by the CNN model can be represented as follows:

$$\hat{p}_d^{t+\Delta t} = f_{CNN}\left(\Psi, \mathbf{u}^*, p_d^t, f_s, c_p, IC\right), \tag{26}$$

where c_p is the network coefficients determined during training process of CNN model as discussed in Section 2.3.1. In Eq. (26), *IC* is the identification character representing a specific class of physical problems, such as *IC* = 'dam' indicating dam breaking problems while *IC*= 'sol' indicating solitary wave propagation. *IC* is specified by users, which does not directly affect the training and output of training in this paper.

2.3.5. Objective function

In the CNN model, the function f_{CNN} is established during the training process by minimising the error between the target and actual pressure, which is represented by an objective function. Different objective functions lead to different measurement of errors and perhaps different prediction of the pressure. Based on our preliminary tests, the objective function f_{obj} used in this work is the sum of squared L-2 norm of the difference between the predicted dynamic pressure and

actual values, which is defined as follows:

$$f_{obj} = \sum_{i=1}^{N} \left(\hat{p}_{d,i} - p_{d,i}\right)^2 , \tag{27}$$

where $\hat{p}_{d,i}$ and $p_{d,i}$ are the predicted dynamic pressure and the actual dynamic pressure at the background node i obtained from the training database. The objective function can also be the sum of weighted squared L-2 norm of the divergence of the predicted velocity as used in Tompson et al. (2017) or the multi-objective loss function including the difference between the target pressure and actual values and the divergence of velocity as in Xiao et al. (2018). In addition, embedding the satisfaction of the physical law (PPE) into the objective function of the network may improve the physical consistency and accuracy of the trained model. However, these have not been tested in this work but may be attempted in our future work.

3. Numerical Study

In this section, two benchmark applications are considered to investigate the performance of the numerical framework for modelling the free surface flow, which will be evaluated by quantitative comparisons of the ISPH_ML results with the experimental data and other numerical results.

3.1. Dam-break flow

The dam-break flow is a classical validation test case for Lagrangian fluid simulations. In this test, a rectangular column of water is confined by a rectangular tank as shown in **Fig. 4**. The width and height of water column are *L* and *H*, respectively. *D* is the length of horizontal section of water tank and a pressure sensor P_1 is located on the right wall at a vertical distance of h_1 from the bottom.

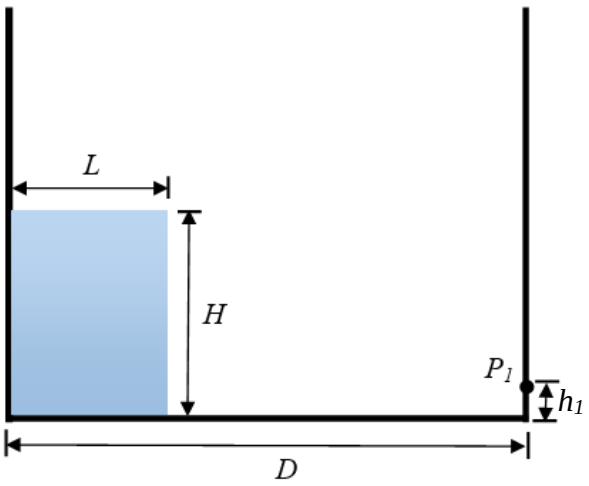


Fig. 4 Schematic view of dam break flow

In order to train the CNN model $f_{CNN}\left(\Psi, \mathbf{u}^*, p_d^t, f_s, c_p, dam\right)$, a series of ISPH_CQ simulations have been carried out. All variables and parameters are non-dimensionalised using *D*. These cases for this purpose are selected in such a way that the lengths and heights of the water column are

randomly specified in the ranges of *L/D* = 0.2~0.5 and *H/D* = 0.2~0.5 with D = 2.0 m. According to the convergent investigation, the initial particle spacing of 0.01 m and a time step size of 0.001 s are used in the ISPH_CQ simulation. It is well understood that the richness of the data in the training and testing sets plays critical role in securing satisfactory accuracy of $f_{CNN}(\Psi, \mathbf{u}^*, p_d^t, f_s, c_p, dam)$. In the preliminary work, training/testing data sets with different number of cases are considered. Fig. 5 illustrates the random variations of a training set of 70 cases/scenes and a testing set of 30 cases/scenes. For each case/scene, 128 frames with 0.006 s time interval are produced for the training. These frames cover the entire process from the initial condition to the occurrence of the violent impact on the right side of the tank.

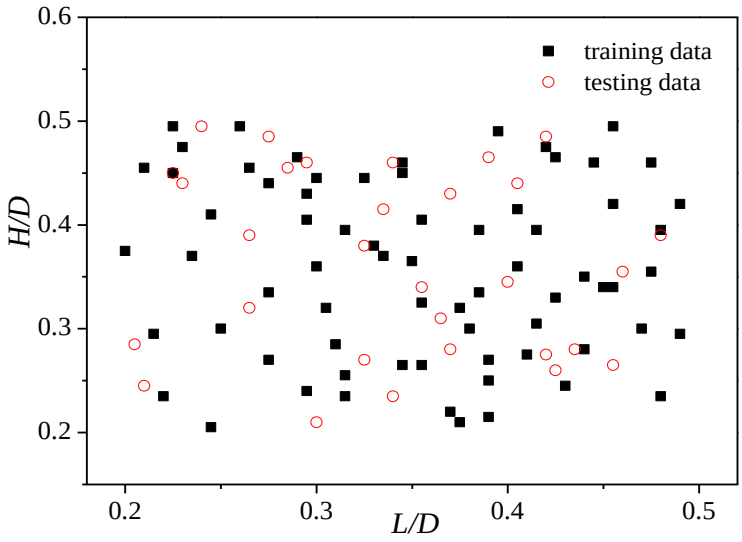


Fig. 5 Variation of the heights and lengths of the water column used in training data set and test data set

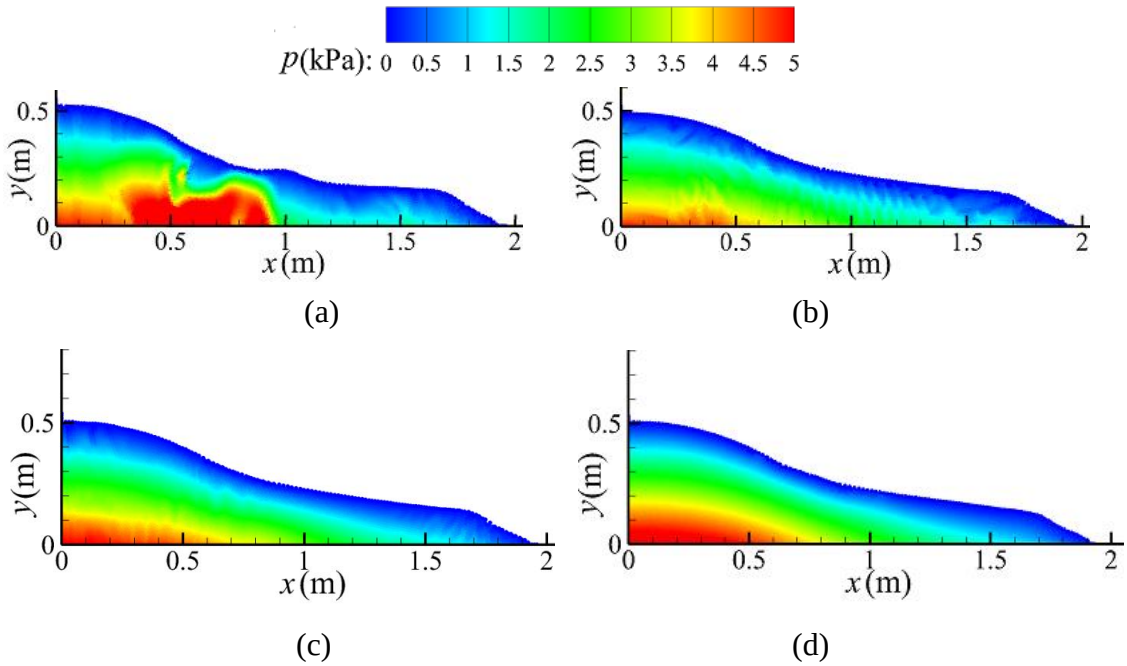
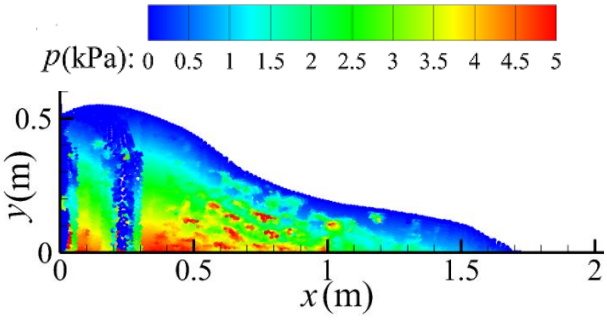
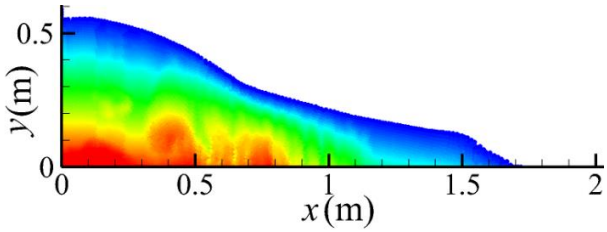


Fig. 6 The particle distributions and pressure contour at t = 0. 45s for the cases with (a) ISPH_ML with 10 scenes in training set and 10 scenes in the testing set, (b) ISPH_ML with 20 scenes in training set and 20 scenes in the testing set, (c) ISPH_ML with 70 scenes in training set and 30 scenes in the testing set, and (d) ISPH_CQ (L= 0.25D, H = 0.5D and D = 2.0 m)

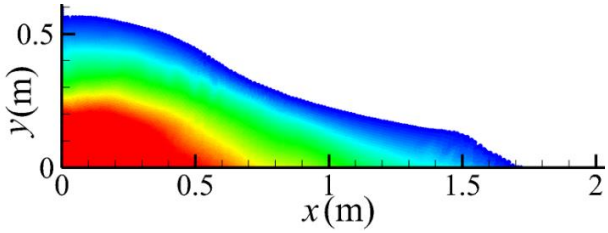
We first discuss how the number of training and testing scenes affect the results. For this purpose, Fig. 6 is plotted, in which the particle distribution and pressure contour at t = 0. 45s for the cases with L= 0.25D, H = 0.5D and D = 2.0 m are compared. These results shown in Fig. 6 (a) – (c) are obtained by using the ISPH_ML in which the dynamic pressure is predicted by $f_{CNN}(\Psi, \mathbf{u}^*, p_d^t, f_s, c_p, dam)$ which is trained/tested using different numbers of scenes, i.e. 10 scenes in the training set and 10 scenes in the testing set (Fig. 6a), 20 scenes in the training set and 20 scenes in the testing set (Fig. 6b), 70 scenes in the training set and 30 scenes in the testing set (Fig. 6c). Fig. 6(d) shows the corresponding ISPH_CQ result for comparison. As can be seen, with the increase of the numbers of scenes in the training/testing data set, the pressure distribution becomes smoother and closer to the corresponding ISPH_CQ result (Fig. 6d). Despite that the accuracy of the ISPH_ML can be further improved by increasing the volume of the database, we will use the test database formed by the cases shown in Fig. 6(c) for further investigation of ISPH_ML behaviours.



(a) Pressure by the trained model with input parameters similar to Tompson et al. (2017)



(b) Pressure by the trained model with input parameters similar to Yang *et al.* (2016)



(c) Pressure by the trained model with input parameters adopted in the present study

Fig. 7 Pressure contour at t = 0. 4s computed by the ISPH_ML with CNN models trained by using different input parameters for the case with L= 0.25D, H = 0.5D and D = 2.0 m

As discussed in Section 2.3.4, the input parameters play an important role on the performance of the trained function f_{CNN}. To shed some light on the improved performance of the input parameters chosen by the present model, the input parameters similar to Yang et al. (2016) and Tompson et al. (2017) have also attempted. The same data sets shown in Fig. 5 are used for three different options. Fig. 7 compares the pressure contour at t = 0.4 s computed by the ISPH_ML with the CNN model trained by using different input parameters for the same case as shown in Fig. 6. In terms of input parameters proposed by Tompson et al. (2017) (Fig. 6a) and those by Yang et al., (2016) (Fig. 7b), the difference is that the latter involves the pressure at the previous time step. Clearly, the pressure distribution shown in Fig. 7(b) is more reasonable and smoother than that shown in Fig. 7(a). This implies the effectiveness of involving the pressure at the previous time step on improving the performance of the CNN model. Compared with the input parameters used by Yang et al. (2016) (Fig. 7b), the present model has additional input parameter $\mathbf{u}^*$. A further improved performance of the result in Fig. 7c than that in Fig. 7b suggests the necessity of involving $\mathbf{u}^*$ as one of the input parameters. It may be worth noting that the input parameters proposed by Yang et al. (2016) and Tompson et al. (2017) may work well with Eulerian projection method and different objective functions. In the Lagrangian framework for the ISPH application, the present model has shown its superiority in all cases considered in our preliminary studies. Due to the length limit of the paper, we only present Fig. 7 for demonstration. It is also worth noting that the trained function $f_{CNN}(\Psi, \mathbf{u}^*, p_d^t, f_s, c_p, dam)$ may not work well at the initial stage due to involvement of $\mathbf{u}^*$ and the pressure at the previous time steps. A simple way to avoid this is to implement the $f_{CNN}(\Psi, \mathbf{u}^*, p_d^t, f_s, c_p, dam)$ after t=T_{ML}, where T_{ML} is specified by users. Before this instant, Eq. (9b) is used to find the pressure like the classic ISPH. Our tests (not present here) shows that results are not very sensitive to the value of T_{ML} as long as it is large enough for building up the sufficient information for $\mathbf{u}^*$ and pressure at the previous time steps.

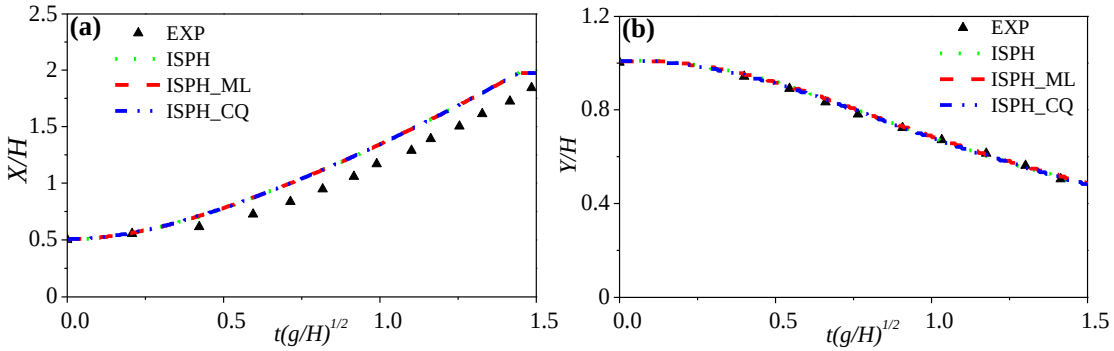
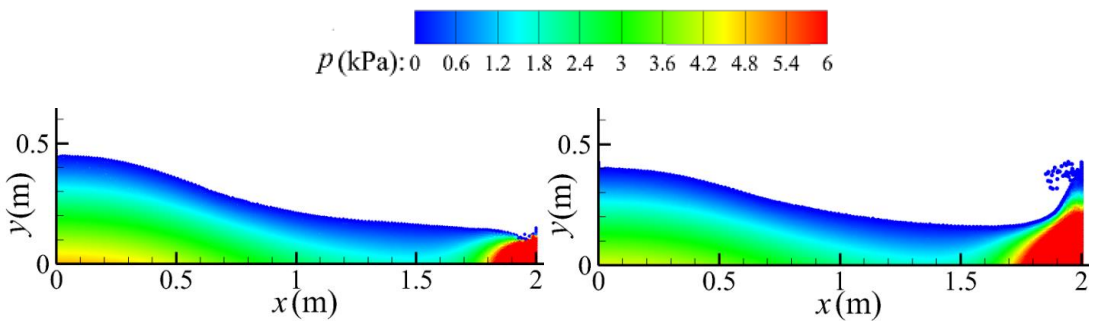
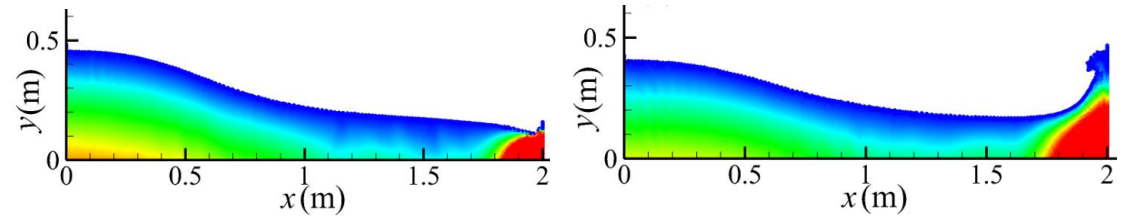


Fig. 8 Comparisons of dam break flow (a) water front; and (b) water column height with experimental data (Martin and Moyce, 1952) in the case with L= 0.25D, H = 0.5D and D = 2.0 m

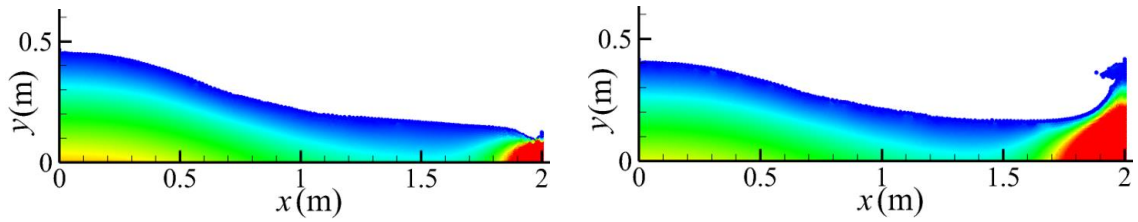
With the above-mentioned preliminary studies, the present ISPH_ML with $f_{CNN}(\Psi, \mathbf{u}^*, p_d^t, f_s, c_p, dam)$ is used and its accuracy and convergence are discussed below. The first case considered is the same as that used in Fig. 6 and Fig. 7, i.e. L= 0.25D, H = 0.5D and D = 2.0 m, which has been experimentally studied by Martin and Moyce (1952). However, the initial particle spacing and the time step sizes are taken as 0.0083 m and 0.0005s respectively, which is different from the parameters for generating the training/test data. One reason for making the difference is for demonstrating that the ISPH_ML can use a wider range of computational parameters once $f_{CNN}(\Psi, \mathbf{u}^*, p_d^t, f_s, c_p, dam)$ is trained. T_{ML} =0.2s in this case. **Fig. 8** displays the wave front and water column height of dam break flow. For the purpose of comparison, the corresponding results of the classic ISPH (Zhang et al., 2018), ISPH_CQ (Zhang et al., 2022) and the experimental data are also plotted. It is observed that the ISPH_ML results agree well with the experimental data, especially the water column height shown in Fig. 8 (b). The ISPH_ML results are almost identical to the corresponding results by ISPH_CQ, which is used to produce the data sets for training the $f_{CNN}(\Psi, \mathbf{u}^*, p_d^t, f_s, c_p, dam)$. The relative difference between the ISPM_ML and ISPH_CQ results of water front and water column height in Fig. 8 are approximately 1%. More discussion about the difference will be given in Section 3.3.



(a) ISPH



(b) ISPH_ML



(c) ISPH_CQ

Fig. 9 The particle distributions with pressure contours at t = 0.5 s (left column) and t = 0.55 s (right column) (L= 0.25D, H = 0.5D, D = 2.0 m and dx = 0.0083 m) obtained by using different ISPH models

Attention is also paid to the pressure field at different time during the simulations. Fig.6 (c) and (d) have illustrated the results at t = 0.45 s, which represents a typical feature of pressure characteristics before the wave impact on the right-end wall, in the case with L= 0.25D, H = 0.5D and D = 2.0 m. More results at different time instants after the occurrences of the impact in the same case are shown in Fig. 9. As observed from Fig.9, the pressure distribution results from the ISPH_ML is almost as smooth as other conventional ISPH results. The free surface location and the formation of the breaking jet in the ISPH_ML simulation are visually comparable to the corresponding results from the ISPH and ISPH_CQ. Quantitatively, the pressure time histories recorded at the pressure monitoring point P1, which is placed on the right end wall and 0.1 m (h_1 = 0.1 m) above the tank bed, are displayed and compared in Fig. 10. The results from all ISPH models exhibit a typical feature of violent wave impact, i.e. a sharp rising of pressure. Taking the ISPH_CQ result as the reference value, the present ISPH_ML slightly overestimate the peak pressure, whereas the classic ISPH underestimate the peak pressure. The relative errors of the ISPH_ML and the classic ISPH in terms of the peak pressure are at a similar level, i.e. approximately 10%, which is acceptable for violent wave impact problems. It is also interesting to observe that there is much less fluctuation in the pressure time history produced by the ISPH_ML.

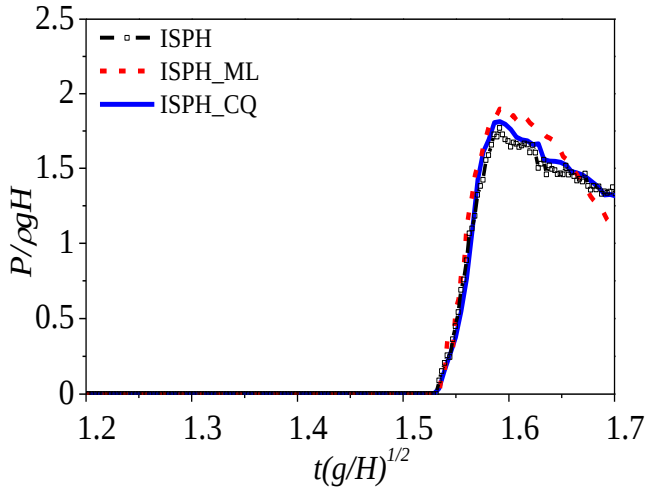
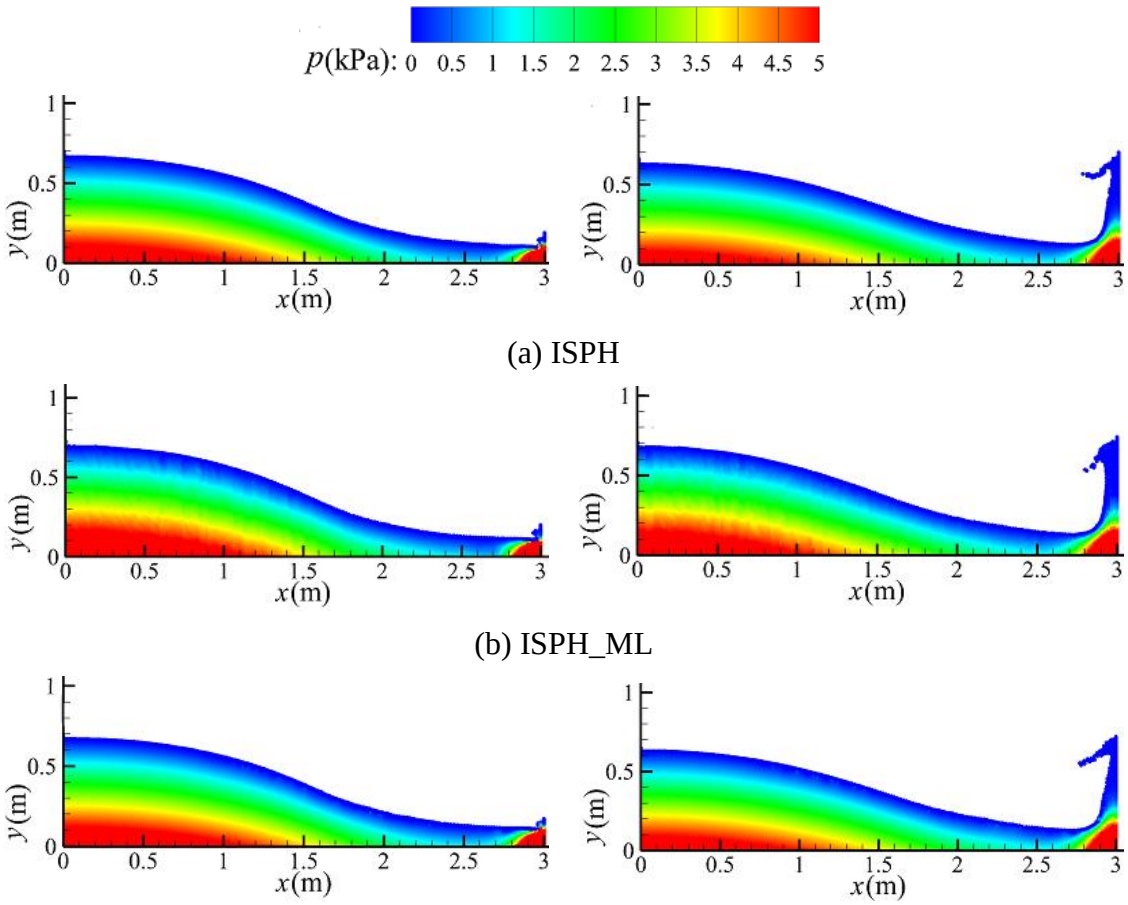


Fig. 10 Time histories of the pressure recorded at P1 in the cases with different ISPH models (L= 0.25D, H = 0.5D and D = 2.0 m; h = 0.1 m)



(a) ISPH

(b) ISPH_ML

(c) ISPH_CQ

Fig. 11 Particle distributions with pressure contours at t = 0.55 s (left column) and t = 0.6 s (right column) (L = 0. 5D, H = 0.25D and D = 3.0 m) obtained by using different ISPH models

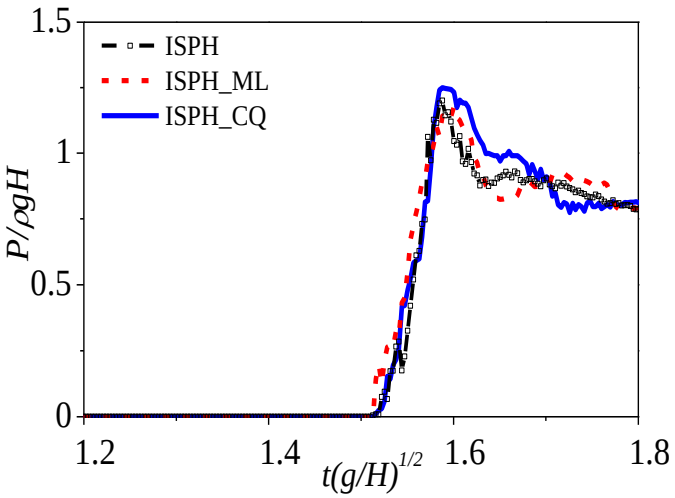
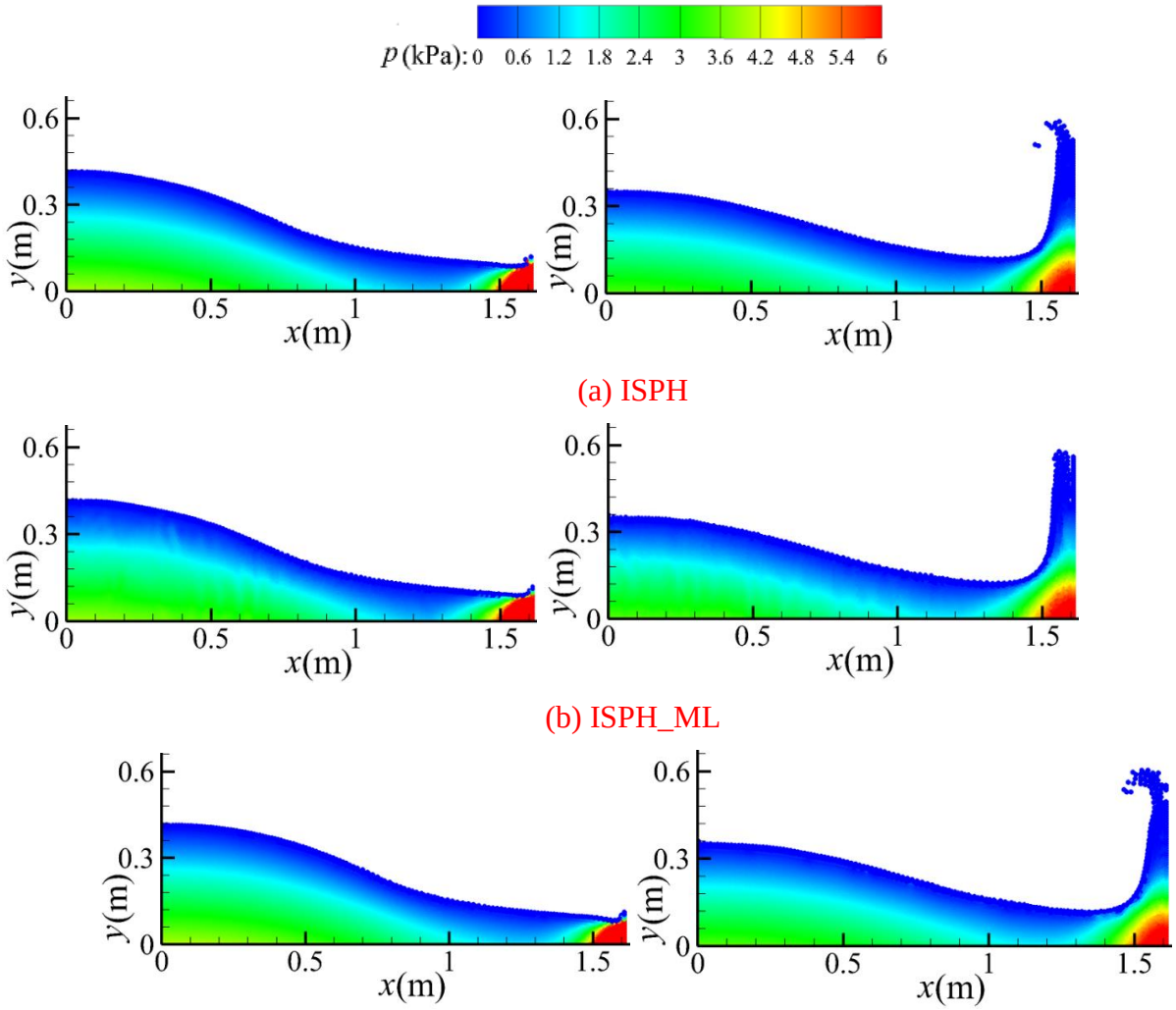


Fig. 12 Time histories of the pressure recorded at P1 in the cases with different ISPH models (L = 0. 5D, H = 0.25D and D = 3.0 m; h_1 = 0.1 m)

The case shown in Figs. 8-10 falls in the range of the training/testing data sets, although it is not identical to any training/testing cases (Fig. 5). Here, the ISPH_ML is applied to another dam breaking cases that is slightly beyond the main range of the training/testing data sets. In this case, L = 0. 5D, H = 0.25D and D = 3.0 m. The particle distributions and pressure contours at different time steps are shown in Fig. 11. Like what has been seen in Fig. 9, the agreements between the ISPH_ML results and the results from other ISPH models are also good. The corresponding time histories of the pressure recorded at P1 are compared in Fig. 12. It is found that the ISPH_ML and the classic ISPH result in similar peak pressure that are slightly lower than the ISPH_CQ result. This implies the present ISPH_ML can deliver satisfactory results for a dam breaking case beyond the main range of training/testing data sets, suggesting it has a potential to be made more general.

In order to further validate the ISPH_ML, another dam breaking case studied experimentally by Lobovský et al. (2014) is considered , where the dimensions $L = 0.37\,D$, $H = 0.37\,D$ and D = 1.61 m are used. The particle size and time step in this case are taken as the same as the case for the results in Fig. 8 and 10. **Fig. 13** illustrates the particle distributions by using different ISPH methods at two instants. Again, one can see that overall particle distributions and pressure fields computed by ISPH_ML as shown in **Fig. 13**(b) are similar to the results from ISPH_CQ in Fig. 13(c). The time histories of pressure at P1 with height h_1 = 0.08 m computed by different ISPH methods are compared with the experimental data of Lobovský et al. (2014) in Fig. 14. The relative error of the ISPH_ML in term of the peak pressure is about 5.3% as compared with the experimental peak pressure.



(a) ISPH

(b) ISPH_ML

(c) ISPH_CQ

Fig. 13 Particle distributions with pressure contours at t = 0.3 s (left column) and t = 0.36 s (right column) (L = 0.37 D, H = 0.37 D and D = 1.61 m) obtained by using different ISPH models

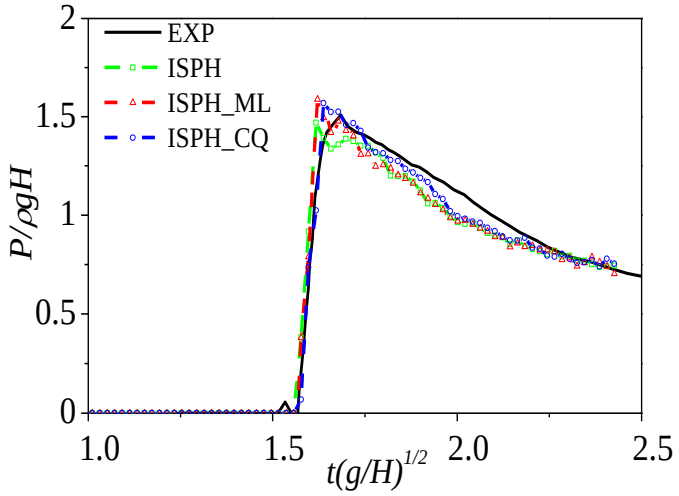


Fig. 14 Time histories of the pressure recorded at P1 in the cases with different ISPH models (L = 0.37 D, H = 0.37 D and D = 1.61 m; h_1 = 0.08 m, dx = 0.0083m)

3.2. Solitary wave propagation

The second benchmark problem considered here is the solitary wave propagation. The schematic diagram of the wave tank to perform the test is shown in **Fig. 16**, in which h is the solitary wave height, D is the water depth and L is the length of the wave tank. The wave is generated by the wavemaker on the left end of the tank using the approach in Ma and Zhou (2009). The right end of the tank is a vertical wall. Before the wave reflected from the right end wall, the free surface can be described by the Boussinesq equation (Lee et al., 1982). The problem has been investigated in our previous work using the classic ISPH and ISPH_CQ (Zhang et al., 2021; 2022).

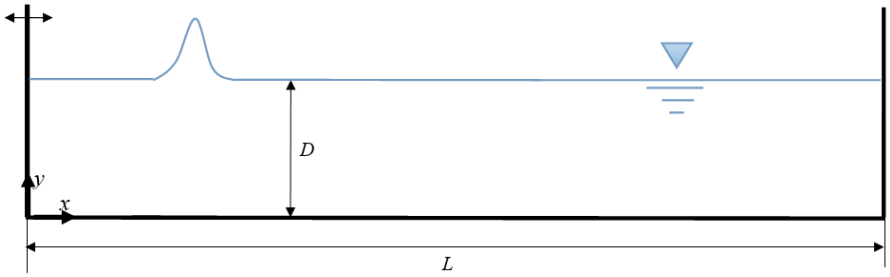


Fig. 16 Schematic wave tank for solitary wave

In this benchmark test, the CNN model $f_{CNN}\left(\Psi, \mathbf{u}^*, p_d^t, f_s, c_p, sol\right)$ is trained using a new set of training/testing data also generated by the ISPH_CQ. The length of the tank is taken as 10 m. All training and testing cases are randomly specified using h ranging from $0.2D$ to $0.4D$, whereas different values of 0.25 m, 0.275 m and 0.3 m are assigned for D. According to the preliminary study, 40 training cases/scenes and 20 testing cases/scenes are used. Each case produces 256 frames of data with a time interval of 0.0125 s. According to the convergent test, the initial particle spacing and the time step size used by the ISPH_CQ to generate the dataset are 0.0125 m and 0.00125 s, respectively. The trained CNN model $f_{CNN}\left(\Psi, \mathbf{u}^*, p_d^t, f_s, c_p, sol\right)$ is then incorporated with the ISPH_ML model for predicting the pressure in simulating the cases with different solitary wave heights. The corresponding results are compared with the other ISPH results and the solution from the Boussinesq equation (Lee et al., 1982).

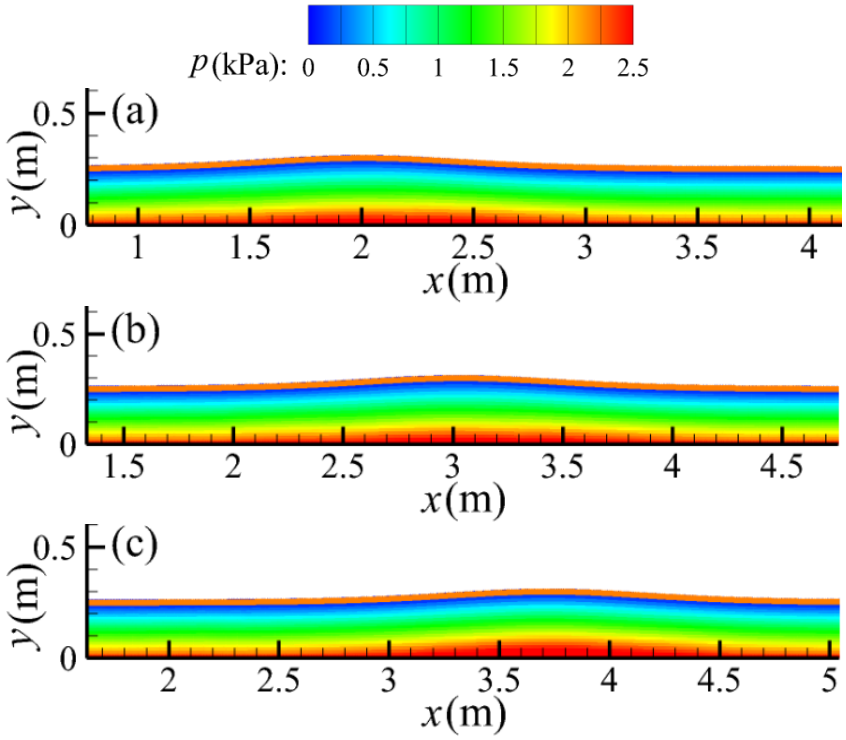


Fig. 17 Particle distributions and pressure contour obtained by the ISPH_ML in the case of solitary wave with $h = 0.2D$ at (a) t= 2.6 s; (b) t= 3.2 s and (c) t= 3.6 s (D = 0.25 m, L = 8 m, the orange line is the results from the Boussinesq equation, Lee et al., 1982)

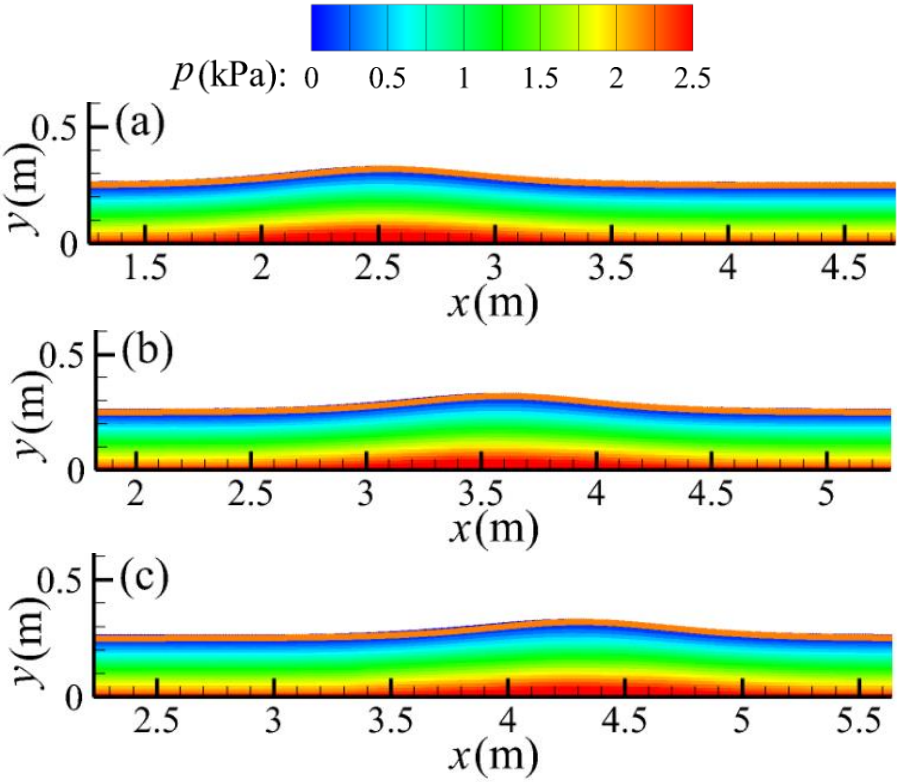


Fig. 18 Particle distributions and pressure contour obtained by the ISPH_ML in the case of solitary wave with $h = 0.28D$ at (a) t= 2.6 s; (b) t= 3.2 s and (c) t= 3.6 s (D = 0.25 m, L = 8 m; the orange line is the results from the Boussinesq equation, Lee et al., 1982)

Fig. 17 and Fig. 18 illustrate the pressure contour obtained by the ISPH_ML in the cases with h = 0.2D and h = 0.28D, respectively. In these cases, the water depth D = 0.25 m and the tank length L = 8 m. For the purpose of comparison, the wave profile predicted using the Boussinesq equation (Lee et al., 1982) are also plotted together (thick line on the free surface). As indicated above, Zhang *et al.* (2021,2022) has compared the numerical results by the classic ISPH and the ISPH_CQ with the Boussinesq solution, and satisfactory agreements have been demonstrated. For clarity, the corresponding results are not shown in Fig. 17 and Fig. 18, but will be displayed in the following section. These figures do not only demonstrate a smooth pressure distribution but also confirms a good agreement in predicting the wave profiles. The relative errors between the ISPH_ML and the Boussinesq solution in terms of the solitary wave height are about 2.8% for h = 0.2D and 4.3% for h = 0.28D. While the relative errors in wave height with h = 0.28D are about 2.3% for ISPH and 1.2% for ISPH_CQ.

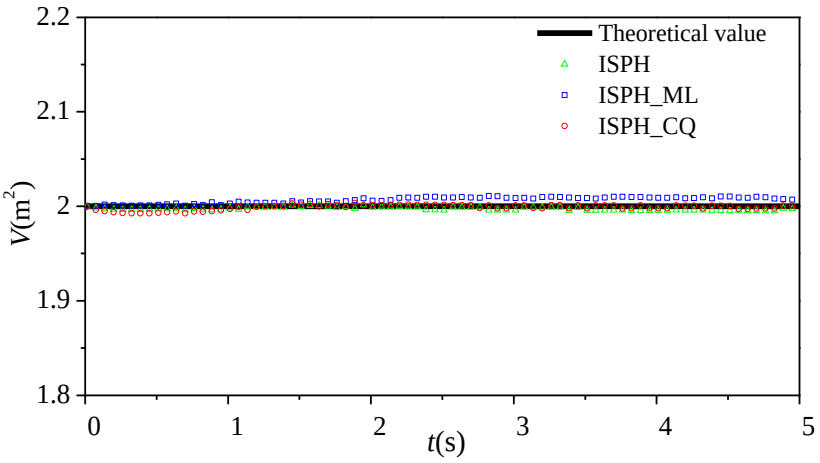


Fig. 19 Time histories of the volume of the fluid in the case of solitary wave (h = 0.28D m and dx = 0.0125 m)

Mass/volume and energy conservation are other important criteria to assess the performance of a numerical method. Fig. 19 depicts the time histories of the fluid volume (equivalent to the mass with a little change fluid density) of the whole domain in the solitary wave case with h = 0.28D. It is observed that ISPH_ ML exhibits a satisfactory performance in volume conservation, although its result is slightly larger than the theoretical value. The maximum error of the volume, defined by $Err_m = \max(|V_i - V_t|)/V_t$ where V_i is the volume values from numerical results at different time step and V_t is the theoretical value, is about 0.53% for ISPH_ML, 0.27% for ISPH and 0.19% for ISPH_CQ in Fig 19. On the other hand, the performances of the ISPH models in terms of energy conservation are also examined. Fig. 20 depicts the time histories of the total energy for the same case as that shown in Fig. 19. In this case, the wavemaker feeds the energy into the fluid and, consequently, the fluid energy increases until the wavemaker stops. During the wavemaker moves, all models give similar results, which increase with the time. After the wavemaker stops, the energy

in the tank is expected to remain as a constant. Taking the ISPH_CQ result from t = 2.0 s to t = 5.0 s in Fig 20 as the reference, the average error of the total energy is about 0.067% for ISPH_ML and 0.046% for ISPH. Generally, the total energy of the fluid in the ISPH_ML simulation is well reserved after the wavemaker stops.

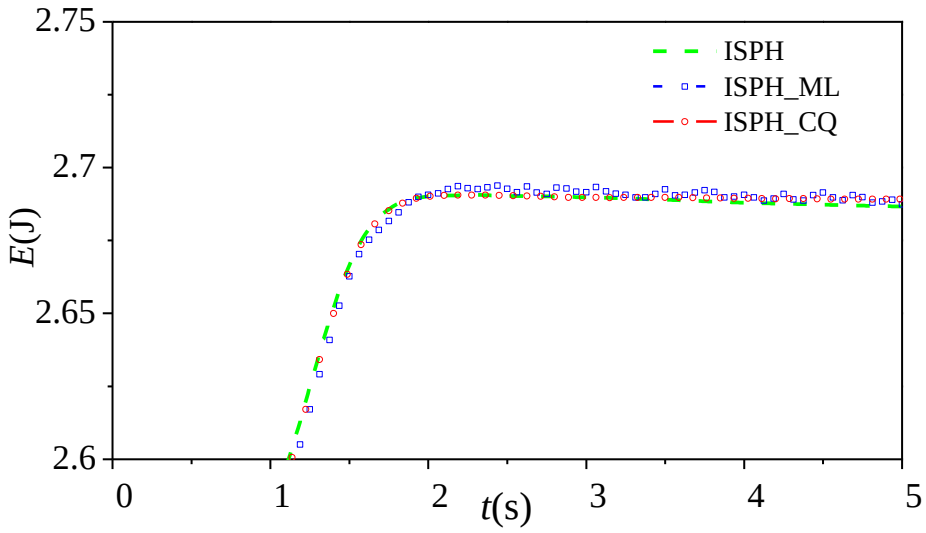


Fig. 20 Time histories of the total energy in the case with solitary wave propagation (h = 0.28D)

3.3. Convergent behaviour and Computational Efficiency

The discussions in Section 3.1 and 3.2 focus mainly on the accuracy and the smoothness of the pressure field. The comparisons shown above confirm a satisfactory accuracy of the ISPH_ML, compared to experimental data, theoretical and other numerical results. In addition, the convergent behaviours and the computational efficiency are extremely important for any numerical method. The existing CNN supported solvers in Eulerian frame by Yang et al. (2016), Tompson et al. (2017), Xiao et al. (2018), Dong et al. (2019), and other machine learning supported particle methods by Ladicky et al. (2015), Wessels et al. (2020), Li and Farimani (2022), all showed the significant improvement of the computational efficiency, but did not investigate the convergence behaviour. Only Li and Farimani (2022) briefly compared the ML results for different time step sizes. In this section, the convergence property of the ISPH_ML and its computational efficiency are discussed.

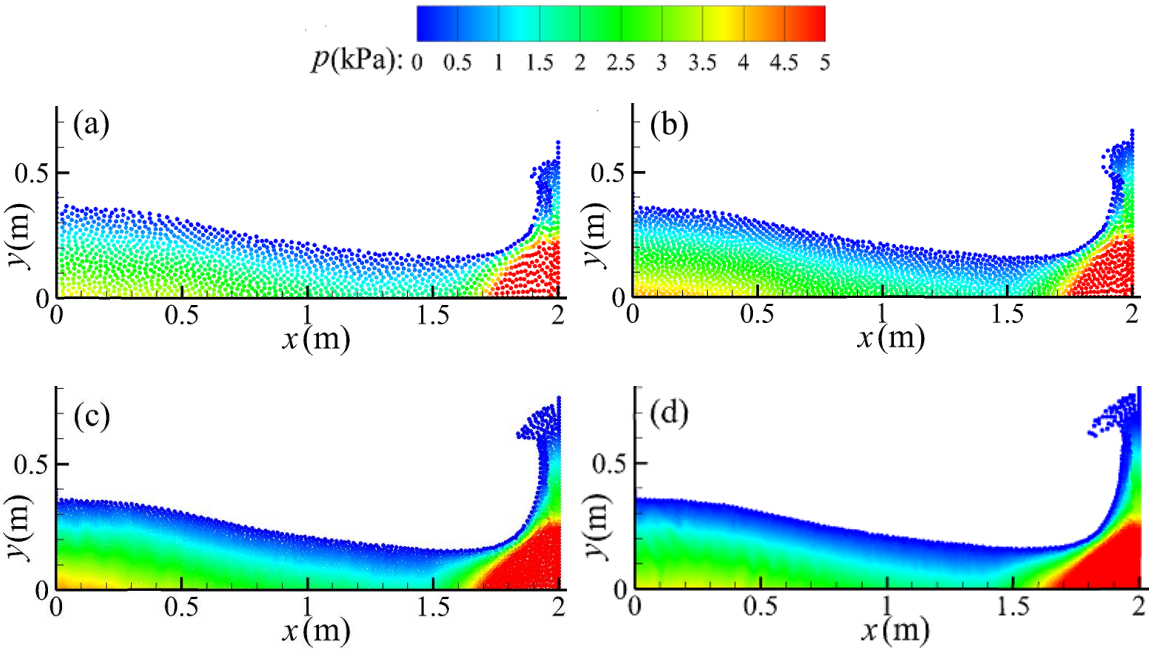


Fig. 21 Particle distributions and pressure contour for a case about dam breaking from ISPH_ML with different particle size: (a) dx = 0.02m; (a) dx = 0.0167m; (c) dx = 0.0125m and (d) dx = 0.01m at t = 0. 6s (L= 0.25D, H = 0.5D and D = 2.0 m)

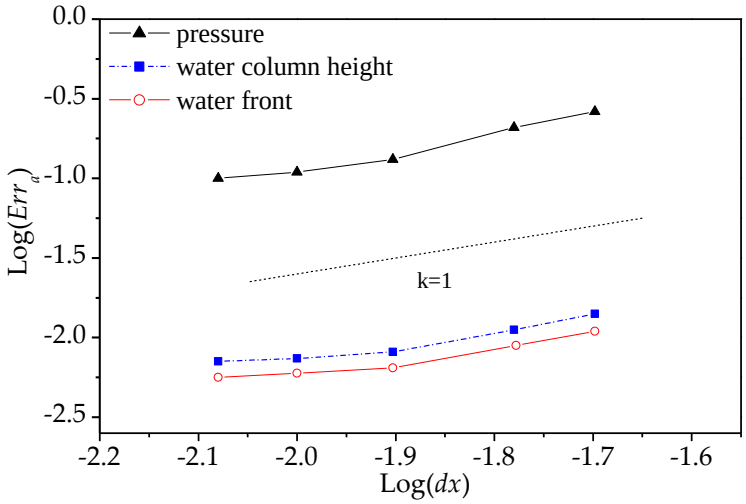


Fig. 22 Averaged errors Err_a (L-2 norm) of numerical results of ISPH_ML in the cases with different initial particle spacings (L= 0.25D, H = 0.5D and D = 2.0 m; ISPH_CQ results are taken as the reference values)

The first case demonstrated here is the dam breaking case shown in Figs. 8 – 10, i.e. L= 0.25D, H = 0.5D and D = 2.0 m. Different initial particle spacings ranging from 0.0083 m to 0.02 m are used. For all these cases, the time step size is determined by $(dt/dx)=(dt_0/dx_0)$, where dt_0 =0.001s and dx_0=0.01m are the time step and initial particle spacing used by ISPH-CQ which has been tested to be appropriate. Fig. 21 illustrates the particle distributions and pressure contour from ISPH_ML with different initial particle spacings. It is found that despite the use of coarse particle resolution, the pressure distribution is reasonably smooth; the free surface profile seems not be different significantly, except for the area near the overturning jet.

Due to limited availability of the experimental data, we use the results from ISPH_CQ simulation with dx = 0.01 m as the reference values. The averaged errors are defined by,

$$Err_a = \frac{\sqrt{\sum_{i=1}^{N}\left(y_{i,n} - y_{i,f}\right)^2}}{\sqrt{\sum_{i=1}^{N} y_{i,f}^{\ 2}}} \tag{28}$$

where $y_{i,n}$ is the results (water column height, water front or the pressure at P1) at different time steps obtained by the present ISPH_ML and the $y_{i,f}$ denotes the corresponding results from the ISPH_CQ simulation, N is the total number of time steps in the duration of simulation, $t\sqrt{g/H}$ = 0.0 to 1.7. The errors of different variables estimated in this way for different initial particle spacing are shown in Fig. 22. As shown by the figure, the averaged errors of the ISPH_ML are reduced as the particle spacing decreases, largely following a linear rate (parallel to the linear sloping line marked 'k=1'), the same rate as the classic ISPH (Zhang et al., 2021; 2022).

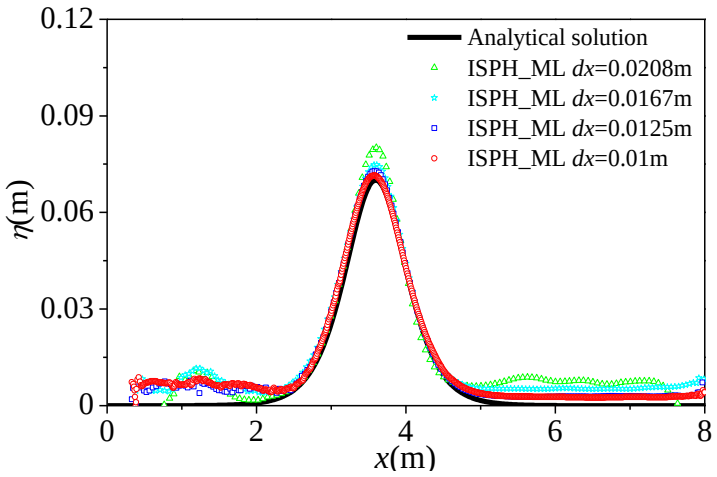


Fig. 23 Comparisons of solitary wave profiles at t = 3.2 s between analytical and ISPH_ML results with initial particle spacings (h = 0.07 m)

In the 2nd demonstration case, we consider the solitary wave propagation with h = 0.28D m (Fig. 18). Different initial particle spacings ranging from 0.008 m to 0.0208 m is used in the convergence investigations with the time step determined by (dx*dt)=(0.00125/0.0125). Fig. 23 illustrates the free surface profiles at t = 3.2 s (Fig. 18 (b)) obtained by ISPH_ML model and the corresponding analytical solution of the Boussinesq equation (Lee et al., 1982). It is observed that as the initial particle spacing reduces, the numerical result becomes closer to the analytical solution. In order to further investigate the convergence property of the ISPH_ML model, the corresponding errors of the free surface profiles with initial particle spacings are given in Fig. 24. The errors in this case are also estimated by Eq. (28) but $y_{i,n}$ and $y_{i,f}$ are taken as the wave elevation at i-th particle recorded from the numerical simulation and that calculated by the Boussinesq equation; N is the total number of particles in the sub-domain specified by x = [2.6 4.6] m as seen in Fig. 23, around the wave crest. For the purpose of comparison, the corresponding results from the classic ISPH and ISPH_CQ are also plotted together. To assist the evaluation of the convergent rate, two slopping lines representing

linear (k = 1) and quadric convergent rates (k = 2) are also plotted. It is found that the ISPH_CQ exhibits a quadric convergent rate when the particle resolution is coarse and reduces to a slower convergence, significantly improved compared with the classic ISPH, which is linear. This further justified the reason why the ISPH_CQ is used to produce the training/testing data sets, in addition to the evidence given in our previous publication (Zhang et al., 2022). More importantly, the ISPH_ML and the classic ISPH show a clear linear convergence rate towards a similar accuracy when the particle spacing is small enough. It firmly concludes that the ISPH_ML with properly trained CNN model can lead to the same accuracy as the classic ISPH.

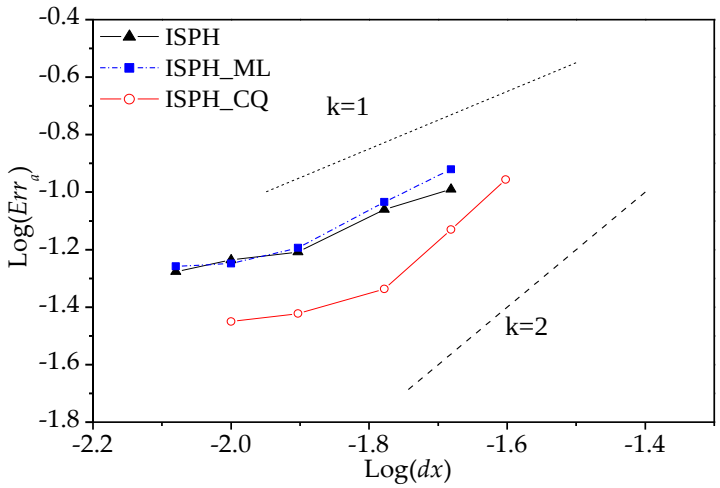


Fig. 24 Averaged errors Err_a (L-2 norm) of numerical results in the solitary wave propagation cases with different particle spacing (h = 0.28D)

Table 1: Case configurations for the solitary wave propagation

Case	D (m)	L (m)	dx (m)	N
1	0.25	10	0.01	25,000
2	0.25	40	0.01	100,000
3	0.25	80	0.01	200,000
4	0.5	80	0.01	400,000
5	0.5	160	0.01	800,000

To show the efficiency of the ISPH_ML, the CPU time spent by the pressure prediction using the $f_{CNN}(\Psi, \mathbf{u}^*, p_d^t, f_s, c_p, sol)$ in the ISPH_ML is compared with that used for directly solving the corresponding PPE in the ISPH model. For this purpose, the case of solitary wave is considered with different computational parameters given in Table 1. All these cases are run on a workstation with Intel Xeon E5-2667 CPU at 3.3 GHz, 16.0 GB RAM and NVIDIA GeForce GTX 1070 with 8 GB of RAM. Fig. 25 compares the average CPU time used in one time step for predicting the pressure using CNN model $f_{CNN}(\Psi, \mathbf{u}^*, p_d^t, f_s, c_p, sol)$ in the ISPH_ML and for directly solving the PPE in the classic ISPH. This figure clearly demonstrates a significant speed-up of the ISPH_ML compared to the classic ISPH as the number of particles is higher than 100k, which is very popular scenario in practices. For example, at 800k particles, the CPU used by the ISPH_ML is about one eighth of that used by the classic ISPH.

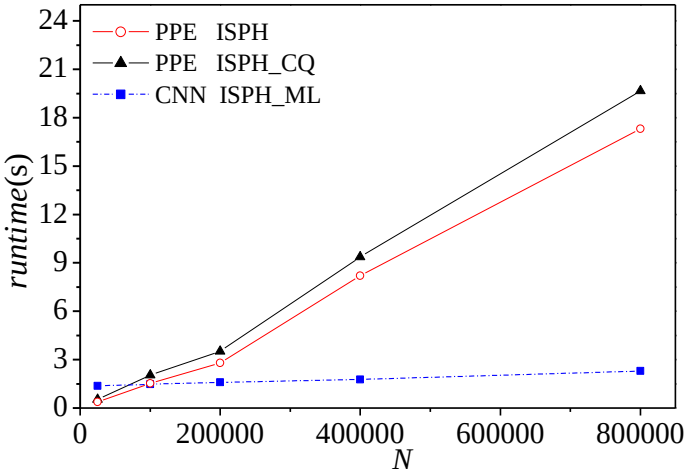


Fig. 25 Averaged CPU time used for pressure prediction per step for the case of solitary wave propagation

4. Conclusion

The paper presents a CNN-supported ISPH model, ISPH_ML, for simulating the free surface flow. In this model, the pressure projection part in the classic ISPH, i.e. solving the PPE, is replaced by the trained CNN model. Although the CNN technique has been widely used in literature, this paper makes several original contributions to combine it with ISPH, including the selection of the input parameters, implementation of the boundary condition, mapping the particle-based field data to grid-structured CNN architecture. To the best of our knowledge, this is the first CNN-supported ISPH model in literature.

The developed ISPH_ML is applied to two classic benchmark problems, i.e. the dam-breaking flow with violent impact and the solitary wave propagation. The accuracy, convergence and the computational efficiency of the ISPH_ML are discussed in detail. It is concluded that (1) the ISPH_ML exhibits a linear convergence, the same as the classic ISPH; (2) both the ISPH_ML and the classic ISPH converge to a similar accuracy; and (3) the particle kinematics and the pressure distribution obtained by the ISPH_ML are visually comparable to the corresponding results by the classic ISPH. Such capacity of ML supported ISPH in reproducing the real physics has not been found in literature, to the best of our knowledge.

More importantly, investigation of the computational efficiency of the ISPH_ML suggests that pressure prediction using the trained CNN model show a significant superiority in terms of CPU time over directly solving the PPE when the number of particles exceeds 100k, which is usually required for modelling violent free surface flow, and that the more particles are involved, the more CPU time is saved.

However, the model bears some limitations and can be improved in future work. For example, only one specific objective function is considered in the paper; other appropriate objective functions, e.g., including the satisfaction of the physical law (PPE), may further improve the generalization and accuracy of the trained model. In addition, this paper has considered only the CNN model; other machine learning models may be attempted, which might be better than the CNN model. Furthermore, this paper attempts only two-dimensional problems; its capacity for three-dimensional problems should be explored.

Acknowledgements

The authors gratefully acknowledge the financial support of EPSRC projects (EP/V040235, EP/T026782, EP/T00424X).

References

Adami, S., Hu, X.Y., Adams, N.A. (2012). A generialised wall boundary condition for smoothed particle hydrodynamics, Journal of Computational Physics, 231, 7057 – 7075.

Bai, J., Zhou, Y., Ma, Y., Jeong, H., Zhan, H., Rathnayaka, C., ... & Gu, Y. (2022). A general Neural Particle Method for hydrodynamics modeling. Computer Methods in Applied Mechanics and Engineering, 393, 114740.

Bai, J., Zhou, Y., Rathnayaka, C. M., Zhan, H., Sauret, E., & Gu, Y. (2021). A data-driven smoothed particle hydrodynamics method for fluids. Engineering Analysis with Boundary Elements, 132, 12-32.

Chow, A.D., Rogers, B.D., Lind, S.J., Stansby, P.K. (2018). Incompressible SPH (ISPH) with fast Poisson solver on a GPU, Computer Physics Communicaitons, 226, 81-103.

Chertkov, M., Tian, Y., Stepanov, M., Fryer, C., Woodward, M., Hyett, C., Livescu, D. (2022). Lagrangian Large Eddy Simulations via Physics-Informed Machine Learning. Bulletin of the American Physical Society.

Cleary, P.W., Monaghan, JJ. (1999). Conduction modelling using smoothed particle hydrodynamics. Journal of Computational Physics,148(1),227-264.

Collobert, R., Kavukcuoglu, K., Farabet, C. Torch7: A matlab-like environment for machine learning. In BigLearn, NIPS Workshop, 2011.

Dong, W., Liu, J., Xie, Z., Li, D. (2019). Adaptive neural network-based approximation to accelerate eulerian fluid simulation. In Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (pp. 1-22).

Fourtakas, G., Rogers, B.D., Nasar, A.M.A. (2021). Towards pseudo-spectral incompressible smoothed particle hydrodynamics (ISPH), Computer Physics Communicaiton, 226,108028.

Fourtakas, G., Stansby, P.K., Rogers, B.D., Lind, S.J. (2018a). An Eulerian–Lagrangian incompressible SPH formulation (ELI-SPH) connected with a sharp interface. Computer Methods in Applied Mechanics and Engineering, 329, 532-552.

Fourtakas, G., Stansby, P.K., Rogers, B.D., Lind, S.J., Yan, S., Ma, Q.W. (2018b). On the coupling of incompressible SPH with a finite element potential flow solver for nonlinear free-surface flows. International Journal of Offshroe and Polar Engineering, 28 (3), 248 - 254.

Garoosi, F., Mellado-Cusicahua, A. N., Shademani, M., Shakibaeinia, A. (2022). Experimental and numerical investigations of dam break flow over dry and wet beds. International Journal of Mechanical Sciences, 215, 106946.

Gotoh, H., Khayyer, A., Ikari, H., Arikawa, T., Shimosako, K. (2014). On enhancement of incompressible SPH method for simulation of violent sloshing flows. Applied Ocean Research, 46(11), 104-115.

Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., Chen, T. (2018). Recent advances in convolutional neural networks. Pattern recognition, 77, 354-377.

Guo, X.H., Rogers, B.D., Lind, S.J., Stansby, P.K. (2018). New massively parallel scheme for incompressible smoothed particle hydrodynamics (ISPH) for highly nonlinear and distorted flow, Computer Physics Communicaitons, 233,16-28.

Hasegawa, K., Fukami, K., Murata, T., Fukagata, K. (2020). Machine-learning-based reduced-order modeling for unsteady flows around bluff bodies of various shapes. Theoretical and Computational Fluid Dynamics, 34(4), 367-383.

Khayyer, A., Shimizu, Y., Gotoh, H., Nagashima, K. (2021). A coupled incompressible SPH-Hamiltonian SPH solver for hydroelastic FSI corresponding to composite structures. Applied Mathematical Modelling, 94(1).

Kutz, J.N. (2017). Deep learning in fluid dynamics. Journal of Fluid Mechanics, 814, 1-4.

Ladicky, Lubor, Jeong, SoHyeon, Solenthaler, Barbara, Pollefeys, Marc, and Gross, Markus. Data-driven fluid simulations using regression forests. ACM Trans. Graph., 34(6):199:1–199:9

LeCun, Y., Boser, B., Denker, J., Henderson, D., Howard, R., Hubbard, W., Jackel, L. (1989). Handwritten digit recognition with a back-propagation network. Advances in neural information processing systems, 2.

Lee, J.J., Skjelbreia, J.E., Raichlen, F. (1982). Measurement of velocities in solitary waves. Journal of the Waterway, 108(2), 200-218.

Li, Z.J., Farimani, A.B. (2022) Graph neural network – accelerated Lagrangian fluid simulation. Computers & Graphics, 103:201 – 211.

Lind, S.J., Xu, R., Stansby, P.K., Rogers, B.D. (2012). Incompressible smoothed particle hydrodynamics for free-surface flows: a generalised diffusion-based algorithm for stability and validations for impulsive flows and propagating waves. Journal of Computational Physics, 231(4), 1499-1523.

Ling, J., Kurzawski, A., Templeton, J. (2016). Reynolds averaged turbulence modelling using deep neural networks with embedded invariance. Journal of Fluid Mechanics, 807, 155-166.

Lobovský, L., Botia-Vera, E., Castellana, F., Mas-Soler, J., Souto-Iglesias, A. (2014). Experimental investigation of dynamic pressure loads during dam break. Journal of Fluids and Structures, 48, 407-434.

Luo, M., Khayyer, A., Lin, P.Z. (2021). Particle methods in ocean and coastal engineering, Applied Ocean Research, 114:102734.

Martin. J. C., Moyce. W. J. (1952). Philosophical Transactions of the Royal Society of London. Series A, Mathematical and Physical Sciences, 244(882), 312-324.

Ma, Q.W. (2008). A new meshless interpolation scheme for MLPG_R method. CMES Computer Modeling in Engineering & Sciences, 23(2), 75-89.

Ma, Q.W., Zhou, J.T. (2009). MLPG_R method for numerical simulation of 2D breaking waves. CMES Computer Modeling in Engineering & Sciences, 43(3), 277-303.

Ma, Q.W., Zhou, Y. and Yan, S.Q. (2016). A review on approaches to solving Poisson's equation in projection-based meshless methods for modelling strongly nonlinear water waves. Journal of Ocean Engineering and Marine Energy, 2(3), 279–299.

Marinho, E.P. (2021). A machine learning approach of finding the optimal anisotropic SPH kernel. In Journal of Physics: Conference Series (Vol. 2090, No. 1, p. 012115). IOP Publishing.

Monaghan, J.J. (2005) Smoothed particle hydrodynamics. Reports on Progress in Physics, 68(8):1703.

Monteleone, A., Burriesci, G., Napoli, E. (2022) A distributed-memory MPI parallelization scheme for multi-domain incompressible SPH, Journal of Parallel and Distributed Computing, 170,53-67.

Wessels, H., Weißenfels, C., Wriggers, P. (2020). The neural particle method–an updated Lagrangian physics informed neural network for computational fluid dynamics. Computer Methods in Applied Mechanics and Engineering, 368, 113127.

Woodward, M. J., Tian, Y., Hyett, C., Fryer, C., Livescu, D., Stepanov, M., Chertkov, M. (2021). Physics Informed Machine Learning of SPH: Machine Learning Lagrangian Turbulence.

Peng, J.Z., Aubry, N., Zhu, S., Chen, Z., Wu, W.T. (2021). Geometry and boundary condition adaptive data-driven model of fluid flow based on deep convolutional neural networks. Physics of Fluids, 33(12), 123602.

Pinaya, W.H.L., Vieira, S., Garcia-Dias, R., Mechelli, A. (2020). Convolutional neural networks. In Machine learning (pp. 173-191). Academic Press.

Raissi, M., Yazdani, A., Karniadakis, G.E. (2020). Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations, Science, 367 (6481) , 1026–1030

Shao, S.D., Lo, E.Y.M. (2003). Incompressible SPH method for simulating newtonian and non-newtonian flows with a free surface. Advances in Water Resources, 26(7), 787-800.

Shimizu, Y., Gotoh, H., Khayyer, A., Kita, K. (2022). Fundamental investigation on the applicability of a higher-order consistent ISPH method, International Journal of Offshore and Polar Engineering , 32(03), 275-284.

Tompson, J., Schlachter, K., Sprechmann, P., Perlin, K. (2017). Accelerating eulerian fluid simulation with convolutional networks. In International Conference on Machine Learning. PMLR, 3424-3433.

Xiao, X.Y., Zhou, Y., Wang, H., Yang, X.B. (2018). A novel cnn-based poisson solver for fluid simulation. IEEE transactions on visualization and computer graphics, 26(3), 1454-1465.

Yan, S., Ma Q.W., Wang J.H. (2020). Quadric SFDI for Laplacian Discretisation in Lagrangian Meshless Methods. Journal of Marine Science and Application 19.3, 362-380.

Yan, S., Ma, Q.W. (2017). A hybrid approach coupling MLPG-R with QALE-FEM for modelling fully nonlinear water waves, proceeding of 27th International Ocean and Polar Engineering Conference, San Francisco, USA.

Yang, C., Yang, X.B., Xiao X.Y. Data driven projection method in fluid simulation. Computer Animation and Virtual Worlds, 27(3-4):415–424, 2016.

You, Y., Khayyer, A., Zheng, X., Gotoh, H., Ma, Q.W. (2021). Enhancement of δ-SPH for ocean engineering applications through incorporation of a background mesh scheme. Applied Ocean Research, 110, 102508.

Zhang, N.B., Zheng, X., Ma, Q.W. et al. (2018). A hybrid stabilization technique for simulating water wave–structure interaction by incompressible smoothed particle hydrodynamics (ISPH) method. J. Hydro-environ. Res. 18:77-94.

Zhang, N.B., Yan, S.Q., Ma, Q.W., Zheng, X. (2021). A QSFDI based Laplacian discretisation for modelling wave-structure interaction using ISPH. Applied Ocean Research, 117, 102954.

Zhang, N.B., Yan, S.Q., Ma, Q.W., Khayyer, A., Guo, X., Zheng, X. (2022). A Consistent Quadric ISPH for free surface flow, submitted to Computers and Fluids.

Zheng, X., Ma, Q.W., Duan, W.Y. (2014). Incompressible SPH method based on Rankine source solution for violent water wave simulation. J. Comput. Phys. 276, 291-314.

Zhang, Z.J., Duraisamy, K. (2015). Machine learning methods for data-driven turbulence modeling. 22nd AIAA Computational Fluid Dynamics Conference (p. 2460).