
Simulating Surface Wave Dynamics with Convolutional Networks

Mario Lino

Department of Aeronautics
Imperial College London
ma11218@ic.ac.uk

Chris Cantwell

Department of Aeronautics
Imperial College London

Stathi Fotiadis

Department of Bioengineering
Imperial College London

Eduardo Pignatelli

Department of Bioengineering
Imperial College London

Anil Anthony Bharath

Department of Bioengineering
Imperial College London

Abstract

We investigate the performance of fully convolutional networks to simulate the motion and interaction of surface waves in open and closed complex geometries. We focus on a U-Net architecture and analyse how well it generalises to geometric configurations not seen during training. We demonstrate that a modified U-Net architecture is capable of accurately predicting the height distribution of waves on a liquid surface within curved and multi-faceted open and closed geometries, when only simple box and right-angled corner geometries were seen during training. We also consider a separate and independent 3D CNN for performing time-interpolation on the predictions produced by our U-Net. This allows generating simulations with a smaller time-step size than the one the U-Net has been trained for.

1 Introduction

We study the application of fully Convolutional Neural Networks (CNNs) to the problem of forecasting surface wave dynamics, the motion of which is described by the shallow water equations [1]. Computational modelling of surface waves is widely used in seismology, computer animation and flood modelling [1, 2]. Our network learnt to simulate a range of physical phenomena including wave propagation, reflection, interference and diffraction at sharp corners. This kind of Neural Network (NN) could supplement or potentially replace numerical algorithms used to solve the shallow water Partial Differential Equations (PDEs), reducing the inference time by several orders of magnitude and allowing for real-time solutions. This has particular relevance in iterative design scenarios and applications such as tsunami prediction.

Our NN produces the physics simulation by performing one time-step each time it is evaluated. The time-step size of these simulations is fixed and must be equal to the one used during training; hence it is not possible to obtain predictions between time points. This inconvenience is common to the vast majority of CNNs used for simulating physics [3, 4, 5]. To overcome this issue, we could train a NN to simulate within a range of time-steps sizes, but, this would imply extensive data sets and training time. As an alternative, we suggest the use of two independent models: a NN for predicting the temporal evolution with a fixed time-step size, and a second NN for performing the time-interpolation on such simulation.

Contribution. We demonstrate that our U-Net architecture is able to accurately predict surface wave dynamics in complex straight-sided and curved geometries, even when trained only on data sets with simple straight-sided boundaries. Our U-Net is able to simulate wave dynamics four orders of magnitude faster than a state-of-the-art spectral/*hp* element numerical solver [6], so it could be

an effective replacement for numerical solvers in applications where performance is critical. We also demonstrate that a 3D CNN is able to time-interpolate our U-Net predictions and increase the temporal resolution of the simulations by a factor of four.

2 Related Work

During the last five years, most of the networks used to predict continuous physics have included convolution layers. For instance, CNNs have been used to solve the Poisson’s equation [7, 8], and to solve the steady Navier-Stokes equations [9, 10, 11, 12, 13, 14]. The evaluation of these networks for prediction is considerably faster than traditional numerical PDE solvers, allowing relatively accurate solutions to be predicted between one and four orders of magnitude faster [9, 12]. For this reason, CNNs are ideal for developing surrogate models, complementing expensive numerical solvers [9, 10], or for real-time animations [4]. Although the authors of earlier work [9, 12, 13] demonstrated the generalisation of their networks to domain geometries not seen during training, these unseen domains contain elementary geometrical entities included within the training data. We go one step further by training the network with exclusively straight boundaries and showing the network is able to generalise to arbitrary fluid domains incorporating boundaries with varying radius of curvature.

To some extent, unsteady physics have been indirectly explored in the field of computer vision [3, 4, 5, 15, 16]. Here, the input to the network is a sequence of past solution fields, while the output is a sequence of predicted solution fields at future times. When predicting unsteady phenomena, there is an additional challenge: keeping the predictions accurate for long time periods. To address this, Kim et al. [4] and others [5, 16] proposed to use encoder-propagator-decoder architectures, whereas Lee and You [3] and Fotiadis et al. [15] continued to use encoder-decoder architectures similar to those used for steady problems. Time interpolation (or frame interpolation) has also been performed by neural networks recently, achieving excellent results in film and video production [17, 18, 19, 20]. We take inspiration from these studies to time-interpolate our physics predictions.

3 U-Net as a Simulation Engine

3.1 Method

Our NN is based on the U-Net architecture [21], which has been extensively used for image-to-image translation tasks [12, 13, 15, 22]. In wave dynamics forecasting, the input sequence and the target share an important amount of information at different length scales, this makes the U-Net a particularly appropriate architecture for our problem. Our U-Net receives six fields as input: the geometry field, Ω , and a sequence of five consecutive height fields, $\{h_s, h_{s+1}, h_{s+2}, h_{s+3}, h_{s+4}\}$. It generates as output a prediction of the subsequent height field, \hat{h}_{s+5} , at the next time point. Hence, each evaluation of the network corresponds to performing a single time-step, and the network is re-fed with past predictions to make further predictions. See Appendix A for more details about our U-Net architecture.

The data sets used during training and testing were generated by solving the inviscid, two-dimensional shallow water equations with Nektar++, a high-order spectral/*hp* element solver [23]. We imposed two forms of boundary conditions: solid wall boundaries, which result in wave reflection and diffraction; and open boundaries, which allow waves to exit the domain. As initial conditions, we considered a *droplet*, represented mathematically by a localised two-dimensional Gaussian, randomly placed inside the domain. Figure 2 shows the seven categories of fluid domains included in the data

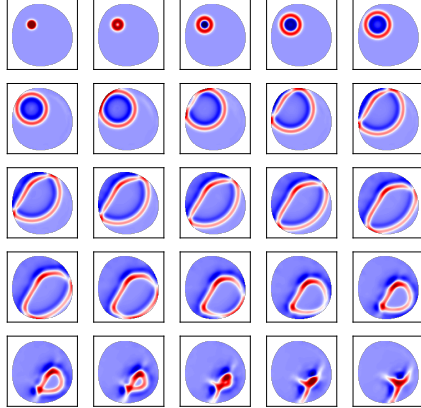


Figure 1: Rollouts of our U-Net. It simulates wave motion on a fluid surface with the possible existence of solid walls [video].

sets. For full details of the data sets, see Appendix B. Our U-Net was trained against the simple closed box and open corner geometries shown in Figures 2a and 2b for five time-steps. The time step was set to $\Delta t = 0.12$ s and the spatial resolution was set to 128 pix/m (details in Appendix A.1)

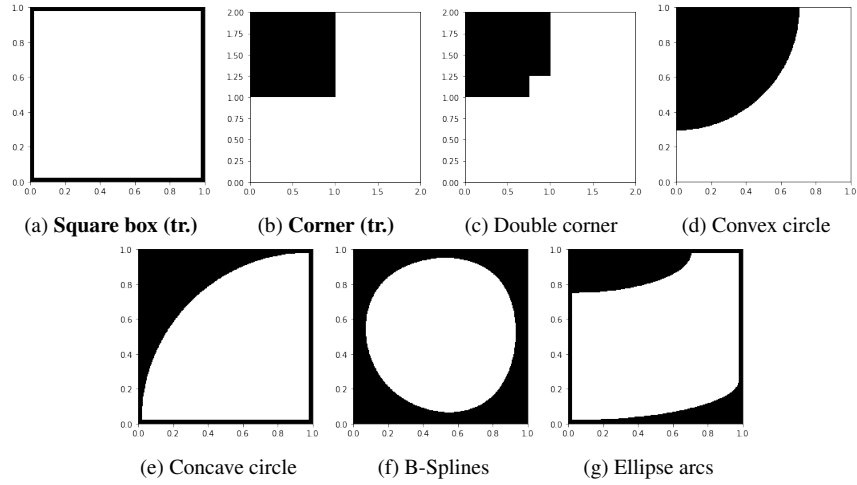


Figure 2: Flow domains on our training (**tr.**) and testing sets. Dimensions in metres.

3.2 Generalisation to Different Domain Geometries

Our network proved to generalise to complex domain geometries not seen during training, some including curved walls, when only the closed domain and open-corner domain were used for training (Figures 2a and 2b). Figure 3 shows that our network is able to accurately predict the wave speed and correctly infer the appearance of reflections at the solid boundaries. The main discrepancy between ground truth and predictions is the height of the wavefronts diffracted by the edges of the steps, which are of lower magnitude in the predicted fields. The reason for this may be that the network is less exposed to wave diffraction than to wave propagation and reflection during training.

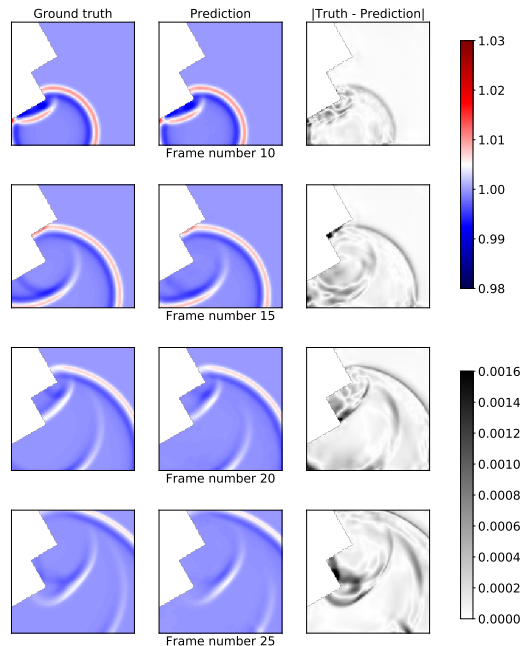


Figure 3: Predictions (left column), ground truth (centre) and absolute difference (right) for a droplet as initial condition [\[video\]](#).

The network is also able to produce good-quality predictions for configurations involving reflections on walls whose radii of curvature is not uniform (see Figure 1), although in these cases more substantial differences between targets and predictions can be seen. The successful generalisation to this kind of domain is likely due to the localised support of the convolution kernels and the architecture. When the convolutions are applied, the network may interpret curved walls as polygonal walls made up of lots of many straight walls of size equal to the stride size. In that case, the higher the image resolution, the smaller the radius of curvature that could be handled by the network.¹

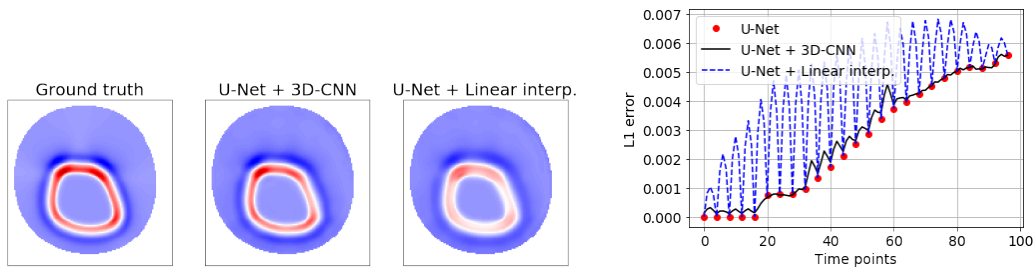
4 3D CNN for Time-Interpolation

4.1 Method

To perform the time interpolation we opted to use a 3D CNN, since 3D convolutional kernels could capture the spatio-temporal dependencies of our problem. We designed our 3D CNN in such a way that the time-step size of the output sequence is four times smaller than in the input sequence, i.e., three temporal frames are inserted between two original temporal frames. Basically, our 3D CNN consists of seven 3D convolutional layers and two transposed convolutional layers (see Appendix C for details). We trained our network with the same simulations used for training the U-Net, but this time adding noise of magnitude similar to the error of the U-Net predictions. The network was trained against simulations with five time-points, returning as output 17 time-points per simulation.

4.2 Results

Figure 4a compares, at a certain time point, the ground truth and the time-interpolations obtained after feeding the U-Net predictions to our 3D CNN and a linear interpolator. We can appreciate an acceptable similarity between the ground truth and the prediction of our (U-Net)-(3D-CNN) model, especially compared to the (U-Net)-(linear interpolator) model. In addition, we can conclude that the discrepancies with the ground truth introduced by our 3D CNN into the U-Net predictions are almost irrelevant. This is evidenced in Figure 4b, where it can be seen that the L_1 error of the U-Net and the L_1 error of the (U-Net)-(3D-CNN) system are of the same order.



(a) U-Net prediction of the wave dynamics (left), time-interpolation of such prediction performed by our network (centre) and linear interpolation (right) [video].

(b) Time point vs L_1 error between the original U-Net predictions, after feeding the 3D-CNN and after linear interpolating.

Figure 4

5 Conclusion

In this work, we investigated the application of fully convolutional deep neural networks for forecasting wave dynamics on fluid surfaces. In particular, we focused on a U-Net architecture with two skip connections, and we trained the network to predict the spatio-temporal evolution of wave dynamics, including: wave propagation, wave interference, wave reflection and wave diffraction. The domains considered during training only included a closed box and a single right-angled corners. However, our U-Net was able to extrapolate to curved walls with varying radii of curvature. The

¹Links to animations comparing the ground truth and the U-Net predictions on the fluid domains in Figure 2 can be found in <https://doi.org/10.6084/m9.figshare.13182623.v1>

RMSE was of order 0.0001 times the characteristic length for at least 20 time steps for both the training and testing datasets. When run on a GPU, these simulations are around 10^4 times faster than the equivalent numerical simulation used for generating our data sets. These findings highlight the potential for neural networks to accurately approximate the evolution of wave dynamics with computational times several orders of magnitude smaller than conventional numerical simulation. Additionally, we proposed to perform deep-learning-based time-interpolation to reduce the time-step size of the simulations, and we used a 3D CNN to accurately increase the number of time points of the U-Net predictions by a factor of four. This approach makes possible to modify the time-step size of the predictions without re-training our physical model.

References

- [1] Mehmet Ersoy, Omar Lakkis, and Philip Townsend. A Saint-Venant shallow water model for overland flows with precipitation and recharge. *arXiv preprint arXiv:1705.05470*, 2017.
- [2] Pilar García-Navarro, J Murillo, J Fernández-Pato, I Echeverribar, and Mario Morales-Hernández. The shallow water equations and their application to realistic cases. *Environmental Fluid Mechanics*, 19(5):1235–1252, 2019.
- [3] Sangseung Lee and Donghyun You. Data-driven prediction of unsteady flow over a circular cylinder using deep learning. *Journal of Fluid Mechanics*, 879:217–254, 2019.
- [4] Byungsoo Kim, Vinicius C Azevedo, Nils Thuerey, Theodore Kim, Markus Gross, and Barbara Solenthaler. Deep fluids: A generative network for parameterized fluid simulations. In *Computer Graphics Forum*, volume 38, pages 59–70. Wiley Online Library, 2019.
- [5] William Sorteberg, Stef Garasto, Alison Pouplin, Chris Cantwell, and Anil A. Bharath. Approximating the Solution to Wave Propagation using Deep Neural Networks. In *NeurIPS Workshop on Modeling the Physical World: Perception, Learning, and Control*, December 2018.
- [6] George Karniadakis and Spencer Sherwin. *Spectral/hp element methods for computational fluid dynamics*. Oxford University Press, 2013.
- [7] Wei Tang, Tao Shan, Xunwang Dang, Maokun Li, Fan Yang, Shenheng Xu, and Ji Wu. Study on a Poisson’s equation solver based on deep learning technique. *2017 IEEE Electrical Design of Advanced Packaging and Systems Symposium, EDAPS 2017*, 2018-Janua:1–3, 2018.
- [8] Ali Girayhan Özbay, Sylvain Laizet, Panagiotis Tzirakis, Georgios Rizos, and Björn Schuller. Poisson cnn: Convolutional neural networks for the solution of the poisson equation with varying meshes and dirichlet boundary conditions. *arXiv*, pages arXiv–1910, 2019.
- [9] Xiaoxiao Guo, Wei Li, and Francesco Iorio. Convolutional neural networks for steady flow approximation. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 481–490, 2016.
- [10] Tharindu P Miyanawala and Rajeev K Jaiman. A novel deep learning method for the predictions of current forces on bluff bodies. In *Proceedings of the ASME 2018 37th International Conference on Ocean, Offshore and Arctic Engineering*, June 2018.
- [11] Emre Yilmaz and Brian German. A convolutional neural network approach to training predictors for airfoil performance. In *18th AIAA/ISSMO multidisciplinary analysis and optimization conference*, page 3660, 2017.
- [12] Amir Barati Farimani, Joseph Gomes, and Vijay S Pande. Deep learning the physics of transport phenomena. *arXiv preprint arXiv:1709.02432*, 2017.
- [13] Nils Thuerey, Konstantin Weissenow, Lukas Prantl, and Xiangyu Hu. Deep Learning Methods for Reynolds-Averaged Navier-Stokes Simulations of Airfoil Flows. *AIAA Journal*, 58(1):15–26, oct 2018.
- [14] Yao Zhang, Woong Je Sung, and Dimitri N Mavris. Application of convolutional neural network to predict airfoil lift coefficient. In *2018 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, page 1903, 2018.
- [15] Stathi Fotiadis, Eduardo Pignatelli, Mario Lino Valencia, Chris Cantwell, Amos Storkey, and Anil A. Bharath. Comparing recurrent and convolutional neural networks for predicting wave propagation. In *ICLR Workshop on Deep Neural Models and Differential Equations*, 2020.

- [16] Steffen Wiewel, Moritz Becher, and Nils Thuerey. Latent space physics: Towards learning the temporal evolution of fluid flow. In *Computer Graphics Forum*, volume 38, pages 71–82. Wiley Online Library, 2019.
- [17] Ziwei Liu, Raymond A. Yeh, Xiaoou Tang, Yiming Liu, and Aseem Agarwala. Video Frame Synthesis Using Deep Voxel Flow. *Proceedings of the IEEE International Conference on Computer Vision*, 2017-Octob:4473–4481, 2017.
- [18] Simon Niklaus, Long Mai, and Feng Liu. Video frame interpolation via adaptive convolution. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017-Janua:2270–2279, 2017.
- [19] Xiongtao Chen, Wenmin Wang, and Jinzhuo Wang. Long-term video interpolation with bidirectional predictive network. *2017 IEEE Visual Communications and Image Processing, VCIP 2017*, 2018-Janua:1–4, 2018.
- [20] Wenbo Bao, Wei Sheng Lai, Chao Ma, Xiaoyun Zhang, Zhiyong Gao, and Ming Hsuan Yang. Depth-aware video frame interpolation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019-June:3698–3707, 2019.
- [21] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham, 2015. Springer International Publishing.
- [22] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.
- [23] Chris D Cantwell, David Moxey, Andrew Comerford, Alessandro Bolis, Gabriele Rocco, Gianmarco Mengaldo, Daniele De Grazia, Sergey Yakovlev, J-E Lombard, Dirk Ekelschot, et al. Nektar++: An open-source spectral/hp element framework. *Computer physics communications*, 192:205–219, 2015.
- [24] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

A U-Net Architecture

The diagram in Figure 5 depicts the U-Net architecture used in the present work. The network receives six fields as input: the geometry field Ω and a sequence of five consecutive height fields $\{h_s, h_{s+1}, h_{s+2}, h_{s+3}, h_{s+4}\}$. The output is a prediction of the subsequent height field \hat{h}_{s+5} . Whereas some recent studies [15, 13] have used bi-linear interpolation to perform the up-sampling, we opted to use transpose convolutions with a 2×2 kernel and stride 2. This increases the number of trainable parameters to 1,864,577, but we also noticed a significant improvement in the quality of the predictions.

A.1 Training

Our U-Net was trained against the simple closed box and open corner geometries shown in Figures 2a and 2b, with a single droplet as initial condition. The time step was set to $\Delta t = 0.12$ seconds and the spatial resolution was set to 128 pix/m. A series of transformations were applied to perform data augmentation and normalisation (see Appendix B.1). We trained for 500 epochs with the Adam optimiser and its standard parameters [24] and the loss function given by

$$\mathcal{L} = (1 - \lambda)\text{MSE}(\hat{h}_i, h_i) + \lambda \left[\text{MSE}\left(\frac{\partial \hat{h}_i}{\partial x}, \frac{\partial h_i}{\partial x}\right) + \text{MSE}\left(\frac{\partial \hat{h}_i}{\partial y}, \frac{\partial h_i}{\partial y}\right) \right] \quad (1)$$

with $\lambda = 0.05$. The learning rate was set to 10^{-4} and decreased by a factor of 10 every 100 epochs. The time origin for the input sequences is not at $t = 0$, instead it is randomly selected, and five time-steps are performed before updating the network weights, whereas the loss function is evaluated after every time-step using backpropagation through time.

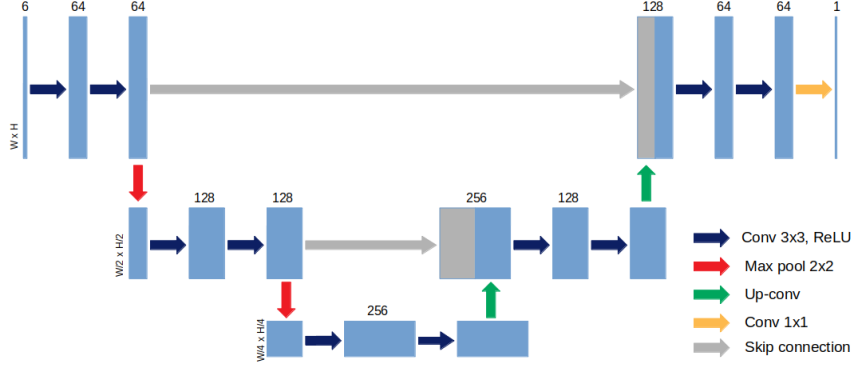


Figure 5: Our U-net architecture with 1,864,577 trainable parameters. It receives six fields as input: the geometry field and a sequence of five consecutive height fields. The output is a prediction of the height field in the subsequent time step.

B Data Sets

All data sets used during training and testing were generated by solving the inviscid, two-dimensional shallow water equations with Nektar++, a high-order finite element solver [23]. In conservative form, these equations are given by

$$\frac{\partial}{\partial t} \begin{pmatrix} h \\ hu \\ hv \end{pmatrix} + \nabla \cdot \begin{pmatrix} hu & hv \\ hu^2 + gh^2/2 & hvu \\ huv & hv^2 + gh^2/2 \end{pmatrix} = \mathbf{0}, \quad (x, y) \in \mathcal{D} \quad (2)$$

where $g = 9.80665 \text{ m/s}^2$ is the acceleration due to gravity and $\mathcal{D} \in \mathbb{R}^2$ denotes the flow domain under consideration. Unknown variables in this system are the water depth $h(x, y, t)$ and the components of the two-dimensional velocity vector $u(x, y, t)$ and $v(x, y, t)$.

We imposed two forms of boundary conditions: solid wall boundaries, which result in wave reflection, diffraction and interference; and open boundaries, which allow the wave to exit the domain. As initial conditions, we considered a *droplet*, represented mathematically by a localized two-dimensional Gaussian superimposed on a unitary depth:

$$h_0^1 = 1 + I \exp\left(-C((x - x_c)^2 + (y - y_c)^2)\right) \quad (3)$$

where I is set to 0.1 m and the values of C , x_c and y_c are randomly selected for each simulation. C follows a uniform distribution such that $C \in (400, 1000) \text{ m}^{-2}$. Similarly, the centre of the Gaussian follows a two-dimensional uniform distribution such that $(x_c, y_c) \in \mathcal{D}$.

The sequences in each data set contain 100 snapshots of the height field sampled at intervals of $\Delta t = 0.03$ seconds. Each sequence is associated with a binary geometry field, $\Omega(x, y)$, which satisfies

$$\Omega(x, y) = \begin{cases} 0, & \text{if } (x, y) \in \mathcal{D} \\ 1, & \text{otherwise} \end{cases} \quad (4)$$

Therefore, $\Omega = 0$ inside the fluid domain and $\Omega = 1$ outside the fluid domain (solid boundaries). The geometry field is an additional input to the network, which provides the required information about the boundaries. Figure 2 shows the geometry field for the seven classes of fluid domains included in the data sets, Table 1 describes these data sets.

The data sets A-C contain only straight-sided fluid domains. Data set A only includes wall boundary conditions, whereas B and C include open boundaries. Data sets D-G contain curved-sided domains. The domains in data set F were generated randomly given four random control points and using B-Splines to create a closed domain. The domains in data set G were also generated randomly given a concave quarter of an ellipse and a convex quarter of an ellipse, whose minor axes follow a uniform distribution between 0.25 and 0.5 m. Only the data sets A and B were used during training. The remaining data sets are used to demonstrate the ability of the network to generalise to arbitrary fluid domain geometries and to two droplets as initial condition.

Table 1: Training and testing data sets.

ID	Dataset	Purpose	Geometry	Initial Condition	Sequences
A	Box_Single_Drop	Training	Figure 2a	Single Drop (eq. (3))	500
B	Corner_Single_Drop	Training	Figure 2b	Single Drop (eq. (3))	500
C	Steps_Single_Drop	Testing	Figure 2c	Single Drop (eq. (3))	200
D	Convex_Single_Drop	Testing	Figure 2d	Single Drop (eq. (3))	250
E	Concave_Single_Drop	Testing	Figure 2e	Single Drop (eq. (3))	500
F	Spline_Single_Drop	Testing	Figure 2f	Single Drop (eq. (3))	200
G	Ellipse_Single_Drop	Testing	Figure 2g	Single Drop (eq. (3))	200

B.1 Normalisation and data augmentation

To improve generalisation across a range of wave dynamics, the height fields were re-scaled according to $\tilde{h} \leftarrow (h - \bar{h}) / (\max(h) - \bar{h})$, where $\bar{h} = 1$ and $\max(h) = 1.1$. This re-scaling was reversed for visualising the network predictions. In order to avoid over-fitting and improve the generalisation capabilities of the network, we apply two sets of transformations to the training data sets. For the data set *A* such transformations consist on random rotations of 90, 180 and 270 deg as well as horizontal and vertical flips. For the dataset *B* random rotations by multiples of 15 deg are applied and the physical size of the frames is reduced to 1 m \times 1 m by cropping the original frames to domains whose center position follows an uniform distribution from 0.9 to 1.1 m in both spatial directions. Finally, the images of all these sequences were linearly interpolated to a 128 \times 128 resolution to satisfy the 128 pix/m requirement. Regarding the testing data sets, sets *C*, *D* are augmented in the same manner as the data set *A*, since they contain open boundaries; and data sets *E*, *F*, *G* are augmented like *B*, as they contain only closed boundaries.

C 3D-CNN Architecture

The 3D CNN used for time-interpolating the physics predictions consists of seven convolutional layers and two transposed convolutional layers with self-normalised ELU activation functions (except the last layer). Table 2 summarises the hyper-parameters of these layers.

Table 2: 3D-CNN layers.

Layer	Type	Output channels	Kernel size	Stride	Padding
1	3DConv	32	3x3x3	1, 1, 1	1, 1, 1
2	3DConv	32	3x3x3	1, 1, 1	1, 1, 1
3	3DTConv	32	2x3x3	2, 1, 1	0, 1, 1
4	3DConv	64	2x3x3	1, 1, 1	0, 1, 1
5	3DConv	64	3x3x3	1, 1, 1	1, 1, 1
6	3DTConv	64	2x3x3	2, 1, 1	0, 1, 1
7	3DConv	64	2x3x3	1, 1, 1	0, 1, 1
8	3DConv	32	3x3x3	1, 1, 1	1, 1, 1
9	3DConv	1	1x1x1	1, 1, 1	0, 0, 0