6.867: Machine Learning
Physics-Inspired Machine Learning for PDEs

Alexis Charalampopoulos
alexchar@mit.edu

Abhinav Gupta
guptaa@mit.edu

Abstract

This work studies the implementation of Machine Learning algorithms for solving nonlinear partial differential equations, in a hybrid framework. Specifically, Recurrent Neural Networks (RNN) and Convolutional Neural Networks (CNN) are used to reduce the high computational costs involved in solving such problems. We do not use Neural Nets to perform predictions for next time-steps but to replace the computationally demanding parts of the system. We implement this hybrid approach for the study of the evolution of free-surface waves. As a first test case, we study the propagation of a wave in a periodic cell. For this case we track the conservation errors regarding the time evolution of the problem. Afterwards we model the reflection of a solitary wave on a vertical wall and compute the force acting on the wall. We also see the effect of varying hyperparameters on the accuracy of prediction of our neural nets, and perform leading order FLOP count comparisons.

1. Literature Review & Motivation

Solving PDE's can be viewed as a non-linear mapping between inputs (initial conditions, boundary conditions, etc.) and output (the desired state variable), which fits very well in the neural network (NN) based machine learning paradigm. Hence, this paper explores the possibility of using various NN based approaches along with some additional physics based understanding of the problems to relieve the computational costs of coupled PDE's.

In general, the PDE's representing physical systems end up being very high dimensional. This is because of the finite dimensional discretization of a infinite dimensional system, which results in a very high computational requirements for their solutions. In an end-to-end machine learning approach, we try to find a non-linear map between potentially very high-dimensional input and output data pairs, which becomes very tough to achieve. In general, it would require a neural network with large number of parameters to be trained with a lot of data. Luckily, for many cases pertaining to the modeling of physical systems, there exists a vast amount of prior knowledge that is currently not being utilized in modern machine learning practice. These include principled physical laws that govern the time-dependent dynamics of a system, or some empirically validated rules or other domain expertise. This prior information can act as a regularization agent that constrains the space of admissible solutions to a manageable size (for e.g., in incompressible fluid dynamics problems the flow should be divergence free in order to respect mass conservation). Hence, using such structured information into a learning algorithm results in amplifying the information content of the data that the algorithm 'sees', enabling it to quickly steer itself towards the right non-linear mapping and generalize well even when only a few training examples are available [8].

The underlying grand goal of using machine learning to solve PDE's is to make it computationally cheaper to solve. Once trained on the data, only a forward pass will be required through the NN to produce the output corresponding to the input, as compared to conventional methods which involve solving a potentially high-dimensional system of linear equations. Conventional methods also suffer from the inability to handle the non-linear parts of a PDE's accurately. The primary interest of the author's of this paper is in fluid systems, which invariably involve high-dimensional linear systems and non-linearities due to intricate coupling between different state variables. In the literature, the intervention of using a NN has been done in many different ways. Some use the end-to-end approach as described earlier, along with incorporating physical conservation laws or the PDE's itself in the loss function. While some authors attempt to directly increase the resolution of coarse simulations by passing it through a NN and creating finer scale structures [3, 11, 13, 14]. But such approaches either require a lot of data or a big NN architecture.

Another prominent approach which has come-up is

a hybrid approach. In most of the numerical solvers, there is a computational bottleneck equation to solve, for example the Laplace equation. Hence in such approaches, only this bottleneck equation is solved using a NN, while otherwise the solver remains the same. Thus accelerating computations, and also reducing the amount of non-linearity the NN needs to capture in such high-dimensional systems [10, 15].

The primary problem of interest studied in this paper is the propagation of irrotational water waves. Order-reduction methods for dynamical systems (e.g. by means of projecting the governing equations) are a staple tool for the study of such physical problems. Such approaches however, fail to yield acceptable predictions for very complicated dynamical systems, due to the large intrinsic dimensionality of the underlying attractor and the possible complexity of transient phenomena. As a result, scientists have started to add "complementary dynamics" to the reduced models, by incorporating time-lagged states of the reduced-order system (up to some reference time origin). The use of delayed states (in an otherwise memoryless physical problem) makes up for the fact that the reduced-order state variable is not enough to exactly represent the dynamics of the problem. The theoretical motivation for this approach stems from the embedding theorems of Takens [9], who proved that the attractor of a deterministic dynamical system can be fully embedded using delayed coordinates.

Hence a NN with some memory would be a natural choice, and thus long-short term memory (LSTM) recurrent neural networks (RNN) could be a good option. Along with LSTM-RNN, Convolutional Neural Network (CNN) or Fully Connected (FC) with the time delays of the state as an input could also work. The training of such NNs takes place by analyzing the difference between the reduced-order model and the data of the full problem (projected to the reduced space). In this project, we propose a study of the effect of using LSTM-RNN and CNN for enhancing the accuracy of reduced order models for high-dimensional irrotational water wave problem. The structure of this project will be as follows. First, we will present a short exposition of the irrotational water wave problem. Next, we will describe the LSTM-RNN and CNN architectures implemented in the hybrid approach for solving our system. Finally, we will study the physical properties of the solutions derived by this hybrid models. Such a study can include the conservation of mass, energy and of other possible invariant quantities, calculation of integral quantities (e.g. force acted on a wall by waves) and L_2 errors compared to simulations of the full problem. We also do leading FLOP count comparisons.

2. Problem Definition

2.1. Irrotational Water Wave problem

The physical problem we shall tackle is the evolution of free-surface waves. Study of water waves and their interactions with the seabed and possible obstacles is a fundamental component of coastal hydrodynamics. The development of fast numerical models that can accurately capture the complex wave dynamics in large domains is essential for many engineering applications at sea. The main model for this problem is the nonlinear potential flow model (NLPF). In this project we aim to reduce the computational cost of using (NLPF) through machine learning. For this reason we will utilize the Hamiltonian formulation of the problem presented in [7]. Since this is a new study, we limit ourselves to the 2D case. For the formulation of the problem, consider a Cartesian coordinate system Oxz, where the x-axis coincides with the quiescent free-surface and the z-axis pointing upwards. An ideal, homogeneous and incompressible fluid fills the time-dependent domain

$$D_h^\eta = \{(x,z) \in X \times \mathbb{R}, z \in (-h(x), \eta(x,t))\} \quad , \quad t \in [t_0, t_1], \tag{1}$$

where $X = [a,b]$ is the common projection of the curves $z = \eta(x,t)$ and $z = -h(x)$ on the x-axis. Under the additional assumption of irrotationallity, the fluid velocity in D_h^η is described by a potential $\Phi = \Phi(x,z,t)$, also called velocity potential. This quantity satisfies the following set of equations (see [6], Ch. 1):

$$\begin{aligned} \Delta\Phi = 0, \quad \text{in} \quad D_h^\eta(X,t) \\ \partial_t\eta + \nabla_x\eta \cdot [\nabla_x\Phi]_{z=\eta} - [\partial_z\Phi]_{z=\eta} = 0 \\ [\partial_t\Phi]_{z=\eta} + \frac{1}{2}[\nabla\Phi]^2_{z=\eta} + g\eta = 0 \\ \nabla_x h \cdot [\nabla_x\Phi]_{z=-h} + [\partial_z\Phi]_{z=-h} = 0. \end{aligned} \tag{2}$$

supplemented with appropriate lateral boundary conditions. We also note that, g is the acceleration of gravirty and $\nabla = (\partial_x, \partial_z)$. Furthermore, boundary values (traces) are denoted by using brackets with a subscript, for example, $[\partial_t\Phi]_{z=\eta} = \partial_t\Phi(x, z = \eta(x,t), t)$. The local depth of the fluid is $H(x,t) = h(x) + \eta(x,t)$ and is assumed to be positive everywhere.

As established in [1], the wave potential can be represented in the form of the following series expansion, realizing an exact semi-separation of variables in the irregular, instantaneous fluid domain $D_h^\eta(X,t)$

$$\Phi(x,z,t) = \sum_{n=-2}^{\infty} \phi_n(x,t) Z_n(z;\eta,h), \tag{3}$$

where Z_n are prescribed vertical basis functions in terms of the local values of the free-surface elevation $\eta(x,t)$ and bathymetry $h(x)$, and ϕ_n are unknown modal amplitudes. Representation (3) establishes the change of functional variables

$$(\eta(x,t), \Phi(x,z,t)) \Longleftrightarrow (\eta(x,t), \{\phi_n(x,t)\}_{n=-2}^{\infty}) \tag{4}$$

which can be employed for the dimensional reduction (elimination of the z variable) of the nonlinear free surface problem (2). Introducing the representation (3) of the wave potential into Luke's variational principle [5], and performing the variations with respect to the new independent functional variables $\eta(x,t)$ and $\phi_n(x,t), n \geq -2$, we eventually obtain, after an extensive analytical treatment presented in detail in [7], the following Hamiltonian evolution equations with respect to $\eta(x,t)$ and $\psi(x,t) = [\Phi]_{z=\eta} = \sum_{n=-2}^{\infty} \phi_n$,

$$\begin{aligned}
\partial_t \eta &= -\partial_x \eta \partial_x \psi + [(\partial_x \eta)^2 + 1](h_0^{-1}\phi_{-2} + \mu_0 \psi) \\
\partial_t \psi &= -g\eta - \frac{1}{2}(\partial_x \psi)^2 \\
&\quad + \frac{1}{2}[(\partial_x \eta)^2 + 1](h_0^{-1}\phi_{-2} + \mu_0 \psi)^2
\end{aligned} \tag{5}$$

and the following system of horizontal differential euqations with respect to the modal amplitude $\phi_n, n \geq -2$ at each (x,t):

$$\begin{aligned}
&\sum_{n=-2}^{\infty} (A_{m,n}\partial_x^2 + B_{m,n}\partial_x + C_{m,n})\phi_n = 0, m \geq -2 \\
&\sum_{n=-2}^{\infty} \phi_n = \psi,
\end{aligned} \tag{6}$$

supplemented by appropriate lateral boundary conditions on ∂X.

The evolution equations (5) are not closed with respect to $\eta(x,t)$ and $\psi(x,t)$ since they contain the free surface modal amplitude ϕ_{-2}. The latter is provided by solving the system of equations at each (x,t). Since the coefficients of this system are defined in terms of $\eta(x,t)$ and $h(x)$ and its excitation is ψ, it is expedient to write

$$\phi_{-2} = \mathcal{F}[\eta, h]\psi, \tag{7}$$

revealing that ϕ_{-2} is in fact a linear, nonlocal operator on ψ, also dependent (nonlinearly) on the boundary functions η and h.

Note that the system is complemented with appropriate lateral boundary conditions depending on the application. Hence, although the dynamical problem can be written for quantities that lie strictly on the free-surface, for the calculation of the modal amplitude ϕ_{-2} we need to solve a 2D problem in the interior of the fluid volume. As a result, it becomes obvious that if we can use Neural Networks to learn the modal amplitudes with respect to η and ψ, we can significantly decrease the computational cost of such simulations.

2.2. Why compute ϕ_{-2} and not η and ψ

We Machine Learn ϕ_{-2} for a few reasons. First, we need to compute fewer variables that way. Furthermore, for η and ψ we would have to do a prediction into the future, i.e. our mapping would be dependent on δt as well. Here, we have to find a mapping between η, ψ and ϕ_{-2} for the same time-instant. We expect that this approach will be much more robust. Furthermore, the computation of ϕ_{-2} allows us to compute quantities like energy. However, if we want to compute velocities, pressure, accelerations in the interior of the fluid, we still have to solve a substrate kinematic problem. In addition, for the evolution equations we do not take spatial derivatives of ϕ_{-2}. As a result, our numerical method is expected to be more robust with respect to spurious effects from the neural net solution. Finally, we would not be able to predict η from modal amplitudes and ψ in the same time step, and computing ψ from ϕ_{-2} and η at the same time step would then require us to take spatial derivatives of ψ making our solution more sensitive to spurious effects.

3. Methodology

3.1. Recurrent Neural Nets - LSTM

RNNs are Neural Networks specifically designed to easily handle dependency of data from a lot of previous time-instances of the input and the state-variable. In theory, RNNs are can account for "long-term dependencies". Yet, in practice generic RNN architectures fail to learn them in any satisfactory manner. The reason behind this inadequacy can be found in [2]. Note: The same problem was also studied by Hochreiter in 1991, but since the manuscript is in German and none of the authors know the language, we were unable to read its results. To fix this shortcoming, Short Term Memory (LSTM) networks were introduced by [4]. LSTMs are explicitly designed to avoid the long-term dependency problem.

For this problem we utilize a standard LSTM cell together with a fully connected layer in order to compute $\{\phi_n^t\}_{n=-2}^{m}$ given η^t and ψ^t. The architecture of the

LSTM approach as well as its coupling with the evolution equations (5) can be seen in fig. 2.

3.2. Convolutional Neural Nets

Convolutional Neural Networks are a big hit in problems where local features are important, and long distance correlations are not important or nonexistent. Such a feature is very prominent in fluid problems, as the motion of a fluid particle is determined by the motion of its neighbouring particles and its current state. Hence CNNs seem to be a natural choice. As argued earlier, adding time-lagged states of the system helps to introduce "complementary dynamics", hence we will include them also in our input. Thus through CNN, we will try to capture both spatial and temporal features to make predictions. Let us assume, that the current time-step is t, and we have access to the last k time-delays for η and ψ. Now we can stack these last k solutions as columns of matrices $[\eta]_{i=t}^{i=t-k} = \begin{bmatrix}\eta^t & \eta^{t-1} & \dots & \eta^{t-k}\end{bmatrix}$ and $[\psi]_{i=t}^{i=t-k} = \begin{bmatrix}\psi^t & \psi^{t-1} & \dots & \psi^{t-k}\end{bmatrix}$. Now our goal would be to have $[\eta]_{i=t}^{i=t-k}$ and $[\psi]_{i=t}^{i=t-k}$ as an input to our CNN, and get $[\phi_{-2}]^t$ as the output. Then we can use $[\phi_{-2}]^t$ to integrate the free-surface evolution equations (5) to get η^{t+1} and ψ^{t+1}. Then again use η^{t+1} and ψ^{t+1} to construct $[\eta]_{i=t+1}^{i=t+1-k}$ and $[\psi]_{i=t+1}^{i=t+1-k}$ and use it as an input for the CNN (see Fig. 1).

3.3. Numerical Implementations

As our first test case, we calculate periodic travelling wave solutions using our hybrid method approach. In more detail, we use a LSTM-RNN and a CNN, for which as input we give the current free-surface elevation η and velocity potential on the free-surface ψ as well as their previous values are preceding time-instances. The output of our Neural Networks will be the value of the modal amplitudes at the current time instant. This way we can immediately integrate in time the free-surface equations and drastically decrease the computation cost of the problem.

3.3.1 LSTM-RNN

For the LSTM-RNN we use one LSTM cell with 30 units and the hidden state with a fully-connected output layer. Assuming a horizontal discretization n_x , the output layer has $4n_x$ nodes while the input layer has $2n_x$ nodes, and the hidden layer $8n_x$ nodes. We set a batch size of 25 and train on a total of 800 time-series of length 150. We train for a total of 100 epochs. Finally, we use Adam optimizer for the training procedure. The weights were initialized using $\mathcal{N}(\mu = 0.01, \sigma = 0.3)$, while the biases using $\mathcal{N}(\mu = 0.0, \sigma = 0.01)$. The loss function consists of the L_2 error of the modal amplitudes. No regularization was needed. We compare our results with numerical solutions derived from a finite-difference code.

3.3.2 CNN

The CNN used to produce all the results, contained one convolutional layer with a filter of 5×3 and ReLU activation. Next comes one pooling layer with 2×2 window and stride 2 max pooling. Then for the fully connected part, we have one hidden layer with $\tanh$ activation. The dimensions of each layer are dependent on the grid resolution of the domain (see fig. 1). The weights were initialized using $\mathcal{N}(\mu = 0.01, \sigma = 0.3)$, while the biases using $\mathcal{N}(\mu = 0.0, \sigma = 0.01)$. For training, Adam optimizer was used. The loss function consisted of L_2 norm of the difference between ϕ_{-2}^t from the NN and that corresponding generated from a numerical solver ($\hat{\phi}_{-2}^t$), and regularization for weights. To prevent overshoots especially near the crest and toughs, max absolute difference between the two was penalized. Atlast, to enforce smoothness in the solution, absolute sum of second order derivative of ϕ_{-2}^t from the NN was added to the loss function. Each of these terms (except the regularization) was divided by the absolute maximum of $\hat{\phi}_{-2}^t$ form the numerical solution. Hence the loss function which needs to be minimized becomes,

$$\begin{aligned} L(\theta) \quad = \quad & \Big(\alpha||\phi_{-2}^t - \hat{\phi}_{-2}^t||_2 + \beta||\phi_{-2}^t - \hat{\phi}_{-2}^t||_\infty \\ & +\gamma \left|\left| \frac{\partial^2 \phi_{-2}^t}{\partial x^2} \right|\right|_1 \Big) / ||\hat{\phi}_{-2}^t||_\infty + \lambda||\theta||_2 \quad (8) \end{aligned}$$

where θ corresponds to weights and biases of the CNN. The relative weightage of each term is controlled by the relative values of α, β, γ, and λ, following the relation $\alpha >> \lambda > \beta \sim \gamma$. The reason behind choosing such relative magnitudes is that, one would like the neural net to first decrease the L_2 error between the prediction and training data as much as possible, and once it is small enough, then focus on making the solution smooth or decreasing the maximum difference. This way we were able to achieve good learning.

4. Results

4.1. Varying the Hyperparameters

We proceed to vary important hyperparameters of the two models and observe their effect on the training and testing error. Training data included waves of different amplitudes and 300 possible different times t. For each amplitude we created 20 time-series with added Gaussian noise where unless specified differently, it was $\mathcal{N}(\mu = 0, \sigma = 0.02)$. For the testing data we used wave

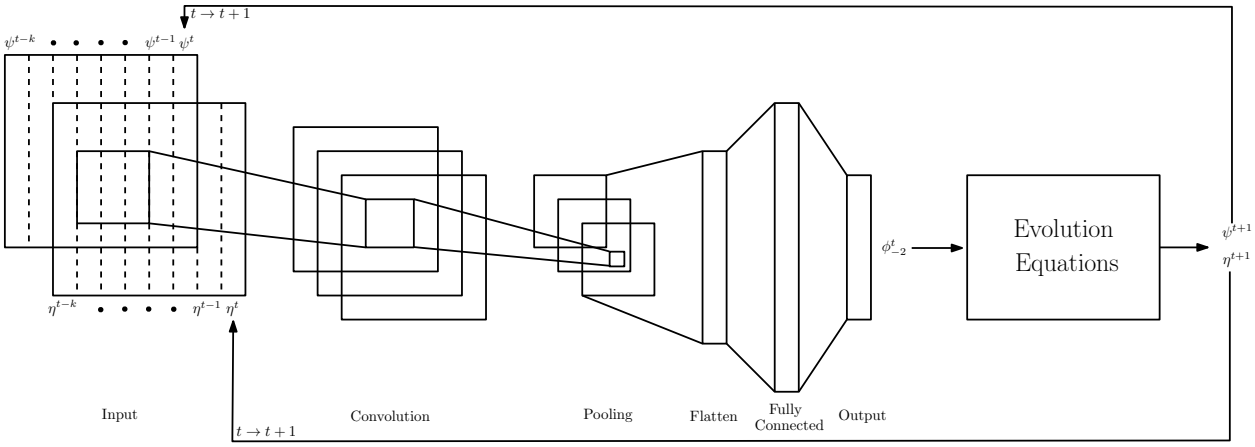

Figure 1. The implemented convolutional neural net (CNN) architecture coupled with the evolution equations (5).

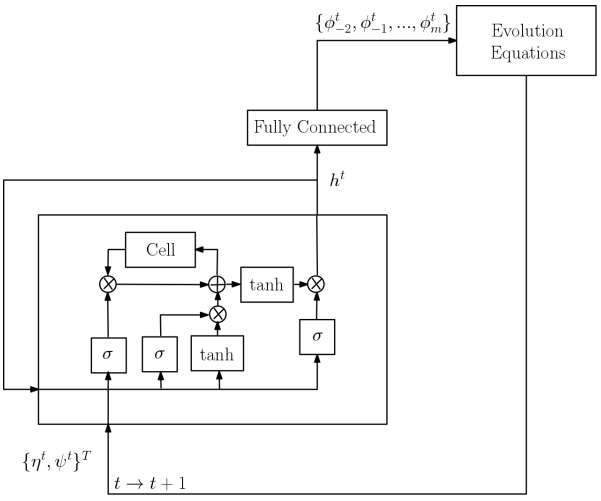

Figure 2. The implemented LSTM-RNN architecture coupled with the evolution equations (5).

amplitudes which were up to 20% higher than those used for training. A number of hyperparameters like learning rate, grid resolution, number of time-delays considered in the input and noise were varied.

For a limited number of epochs (100, with a batch size of 25 and total of 32 batches in each epoch), having a very small learning rate slows down convergence and one would need a large number of epochs to reach some minima. Thus, because of having a limited number of epochs, we are not able to converge much and hence the high training and test errors (fig. 3). Even having a very large learning rate does not help, as one would probably overshoot minima and might not converge or even diverge. Hence in general there exists a sweet spot in between, which is very much prominent in fig. 3 especially for the CNN, while for LSTM-RNN we will requite to go even lower. For this case, the optimal learning rate lies around 10^{-3} for CNN, and $10^{-4} - 10^{-5}$ for LSTM-RNN (fig. 3).

Next we see the how learning is affected by increasing grid resolution. The governing equations involve partial derivatives w.r.t to space and time, which are essentially local features and motivated the use of CNN. One could imagine that the filter could possibly be representing the action of differentiation. As we increase the grid resolution, such derivatives become more and more accurate. Hence one would expect the performance of the CNN to improve with increase in resolution. Also at the same time, the number of weights in the architecture also scales up with the grid resolution. Thus, one would expect the training and test error to rapidly decrease at first and then level-out while further increasing the resolution. This trend is more-or-less followed in fig. 3. As there is not much improvement after 61 grid points, hence we use it in all other results for CNN. On the other hand, LSTM-RNN appears to be more robust to changing grid resolution (fig. 3).

We have also investigated the effect of introducing noise to the training data. We add standard normal Gaussian noise to normalized free surface elevation (η) and free surface potential (ψ) (normalized respectively with their maximum value present in the data) while training. One would expect increase in both training and testing error on increasing the standard deviation of the noise, which is also depicted in fig. 3. However, adding noise allowed us to avoid overfitting and thus increase the accuracy of prediction in time.

The last hyperparameter which we investigated was the number of time-delays considered in the input. For both CNN and LSTM-RNN, error increases if we take less number of time-delays, which makes sense because we are adding less amount of "complementary dynam-

ics". On increasing the number of time-delays the errors for LSTM-RNN flattens out as after a point the LSTM just ignores the additional time-delays used. But CNN appears to overfit after a certain number of time-delays with the testing error increasing as seen in fig. 3.

[12]

4.2. Coupling

For the next step, we couple our trained neural net architectures with the free surface evolution equations (5). We test our hybrid model on the propagation of regular waves on a periodic cell. We pick one of the waves which we used for testing in the previous subsection, and evolve it in time for one period. Throughout the simulation, we track the error in the conservation of mass, momentum and energy. Results of the simulations are presented in fig. 4. For time $t = 2.3524 = T/2$ (which corresponds to half the period), both models do a good job at approximating the free surface elevation. However, for $t = 4.7049 = T$ (which corresponds to one period), both methods have deviated from the numerical solution, presenting a non-smooth free surface elevation.

In order to track the performance of both the methods over time for wave propagation, we track their ability to preserve physical quantities like mass (M), energy (E) and momentum (P). For the initial stage of the evolution, LSTM does a pretty good job in preserving these quantities compared to CNN. However in the long run, both of them produce significant errors in the conservation of these quantities, making long-time simulations difficult for the current setup. The larger memory bandwidth of LSTM-RNN maybe the reason why it is better able to perform better in the initial stages of the evolution as compared to CNN.

4.3. Solitary Wave

Now we turn our attention to the physical problem of reflection of a solitary wave over a vertical impermeable wall. The configuration of the problem is presented in fig. 5. We train on data for reflection of solitary waves of amplitude $a/h_0 = 0.05 : 0.025 : 0.225$, where h_0 is the depth of the domain. We test our LSTM model for solitary wave of amplitude $a/h_0 = 0.25$. We use horizontal discretization $n_x = 101$. For this problem, we assume the time-series of η and ψ given. We now seek to compute inner fluid quantities like pressure and velocities, as well as the force acting on the wall due to the wave. Results are presented in fig. 6. We use two metrics to measure the accuracy of our hybrid approach in this case. The first one is the comparison of the time-series of the force acting on the wall (F_w), compared with the results produced by a numerical PDE solver [7]. For this quantity, we see a very good match with the high accuracy results provided by the PDE solver. For the second metric, we utilize the fact that the free surface is a material surface, and thus cannot take any load. As a result, assuming constant atmospheric pressure above the free surface (which can always be set to 0 with appropriate rescaling), we know the pressure on the free surface should exactly be zero. Hence we measure the large deviation from this value which is 0.02 for the normalized pressure ($P_{surf}/\rho g h_0$), where ρ is the density of the water and g is the acceleration of gravity. This metric is also adequately satisfied showcasing the model's ability to mimic the physics of the problem.

4.4. Computational Cost Comparison

Now we will compare the computational cost of solving the classical formulation (Laplacian equation (2) + evolution equations (5)) with a hybrid formulation (Neural Net + evolution equations (5)). As the NN part was coded in Python, while the original numerical solver was implemented in C, hence direct comparison of computational costs is not possible. Thus we will compare the FLOPS required to solve the Laplacian equation for the velocity potential in a 2D domain, with a feed-forward pass of the implemented NNs. Let us assume that the 2D domain is discretized into n_x number of points in the horizontal (x) direction, and n_z number of points in the vertical (z) direction. Then solving the Laplacian equation (2) using LU-Decomposition would incur a FLOP count which scales asymptotically as $O(n_x^3 n_z^3)$. While FLOP count for feed-forward pass for both CNN and LSTM-RNN would scale as $O(k n_x^2)$, considering k time-delays used. Hence once trained, the NN implementation will decrease the computational requirements significantly.

5. Conclusions/Future Work

In this work, we investigated the use of neural nets (NNs) for solving the water wave problem in a hybrid approach. In this hybrid approach, we replace the most computationally expensive part of the solver with a NN, while keeping the other parts the same. Overall neural nets appear to be promising for solving PDE's and a future viable alternative to conventional numerical solvers. This is mainly due to their ability to bring down the FLOP count requirements. Both LSTM-RNN and CNN implementations had similar performance in terms of approximating the evolution of free-surface, but CNN performs worse in preserving physical quantities like mass, momentum and energy over time. It was also shown that NNs can also be used to compute quantities in the interior of the fluid by using data only at the free surface. An interesting observation was that, introducing noise while training actually improved the ro-

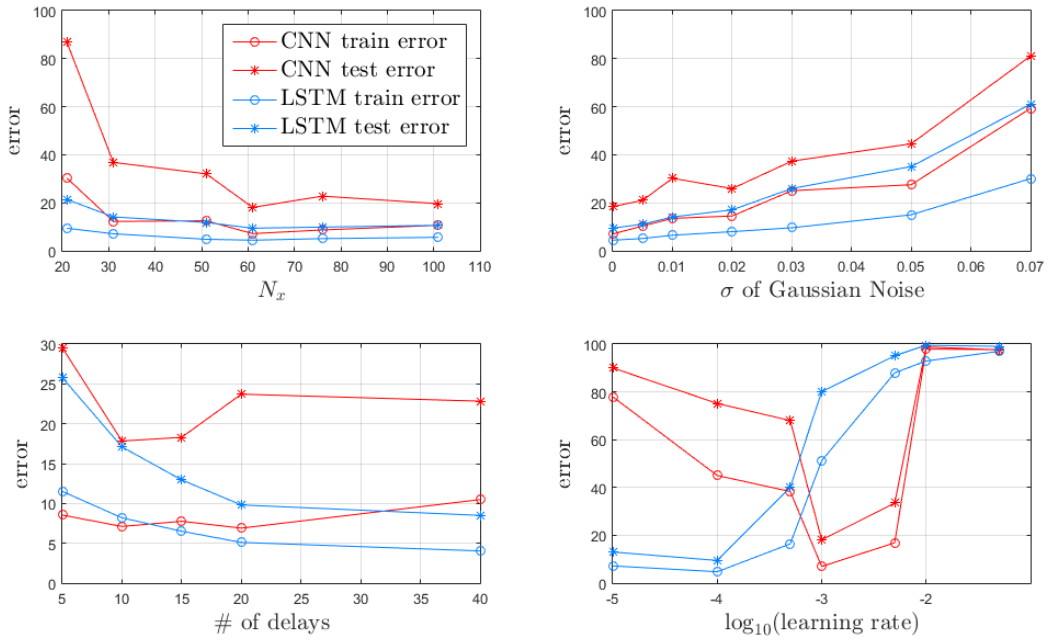

Figure 3. Effect of varying hyperparameters on the testing and training error for both CNN and LSTM-RNN implementations.

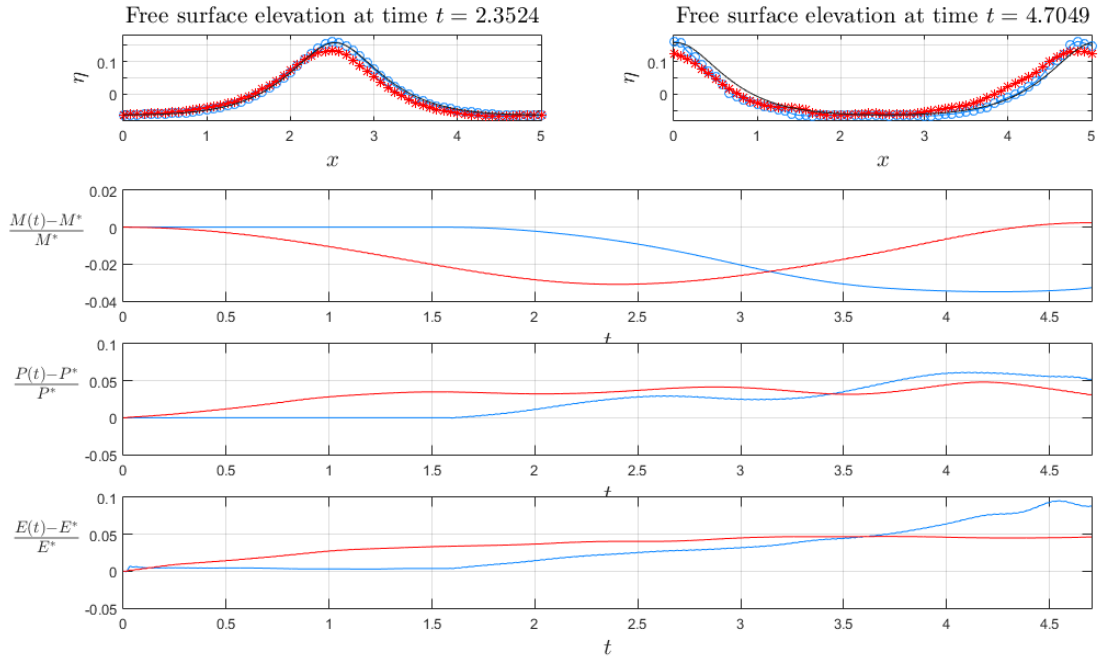

Figure 4. Wave propagation using both CNN and LSTM-RNN. The top two plots shows the snapshots of free surface elevation at two different time instants. The next three plots capture the conservation error of mass (M), momentum (P) and energy (E) with time. Red color corresponds to CNN and blue to LSTM-RNN.

bustness of our while evolution in time. However, more research needs to be done to improve the long time accuracy of our models.

Due to similar level of performance, in the future

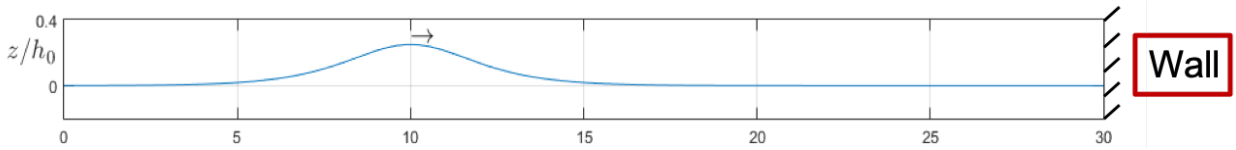

Figure 5. Configuration for reflection of solitary wave over a vertical wall.

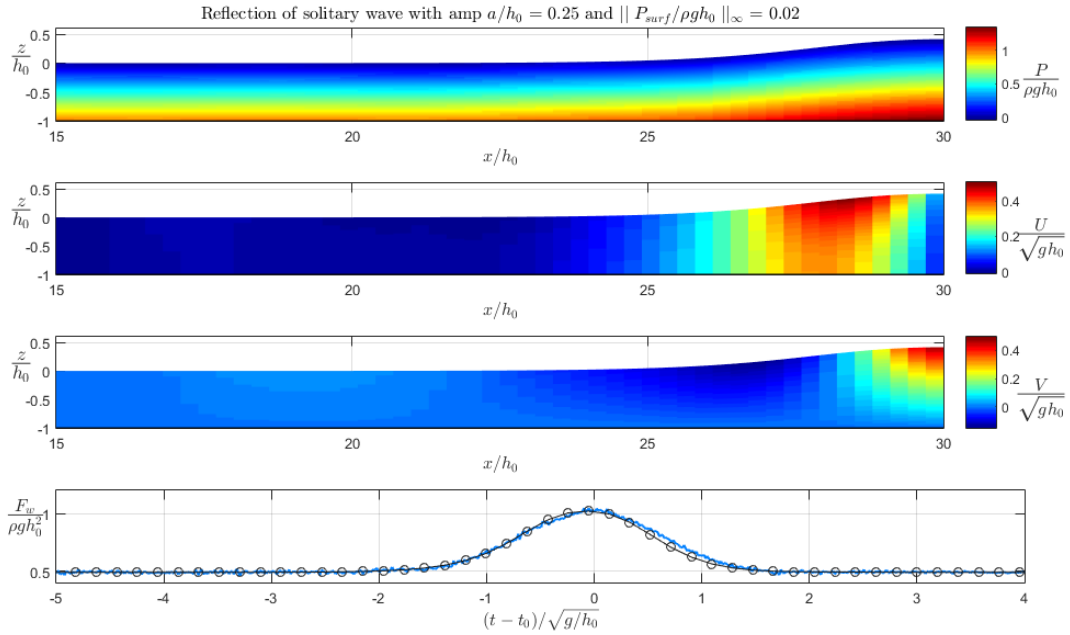

Figure 6. Reflection of solitary wave due to a vertical, impermeable wall on the right. The top three plots depict the normalized pressure, horizontal velocity and vertical velocity respectively. In the bottom-most figure, we plot the variation of force acting on the wall with time. The blue line corresponds to the LSTM prediction and the black circles to a numerical PDE solver.

we can combine the property of LSTM for long-term memory retention and further reduce the FLOP count by stacking it with a convolutional layers. Also incorporating bathymetry in the input of NNs should improve their performance and especially help when we have non-flat bathymetry.

6. Work distribution

Both the authors equally contributed to the extensive literature survey. The LSTM-RNN model was implemented by Alexis, while CNN model by Abhinav. Also both the authors had equal contribution in making the presentation and writing this manuscript. We would also like to acknowledge the contribution of Shashank Agarwal in the initial literature survey when we were selecting the topic of the project.

References

[1] G. Athanassoulis and C. Papoutsellis. Exact semi-separation of variables in wave guides with nonplanar boundaries. *Proc. R. Soc. Lond. A.*=, 473, 2017.

[2] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

[3] M. Chu and N. Thuerey. Data-driven synthesis of smoke flows with cnn-based feature descriptors. *ACM Transactions on Graphics (TOG)*, 36(4):69, 2017.

[4] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[5] J. Luke. A variational principle for a fluid with a free surface. *Journal of Fluid Mechanics*, 27, 1967.

[6] C. Mei, M. Stiassnie, and D. Yue. Theory and applications of ocean surface waves. 2005.

[7] C. Papoutsellis and G. Athanassoulis. A new efficient hamiltonian approach to the nonlinear water-wave problem over arbitrary bathymetry. *Journal of Fluid Mechanics*, 2017.

[8] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics informed deep learning (part i): Data-driven solutions of

nonlinear partial differential equations. *arXiv preprint arXiv:1711.10561*, 2017.

[9] F. Takens. Detecting strange attractors in turbulence. In *Dynamical systems and turbulence, Warwick 1980*, pages 366–381. Springer, 1981.

[10] J. Tompson, K. Schlachter, P. Sprechmann, and K. Perlin. Accelerating eulerian fluid simulation with convolutional networks. *arXiv preprint arXiv:1607.03597*, 2016.

[11] K. Um, X. Hu, and N. Thuerey. Liquid splash modeling with neural networks. In *Computer Graphics Forum*, volume 37, pages 171–182. Wiley Online Library, 2018.

[12] Z. Y. Wan, P. Vlachas, P. Koumoutsakos, and T. Sapsis. Data-assisted reduced-order modeling of extreme events in complex dynamical systems. *PloS one*, 13(5):e0197704, 2018.

[13] S. Wiewel, M. Becher, and N. Thuerey. Latent-space physics: Towards learning the temporal evolution of fluid flow. *arXiv preprint arXiv:1802.10123*, 2018.

[14] Y. Xie, E. Franz, M. Chu, and N. Thuerey. tempogan: A temporally coherent, volumetric gan for super-resolution fluid flow. *arXiv preprint arXiv:1801.09710*, 2018.

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