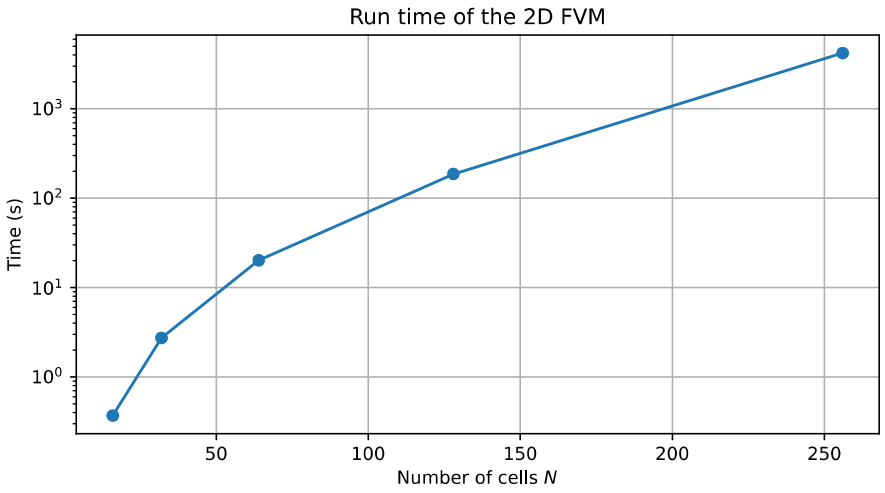

Figure 4.13: Scalability of the FVM to solve the 2D SWE.

From Table 4.3 and Figure 4.13 we see that the run time increases drastically with the number of cells N. This is expected, as the number of cells increases, the number of computations increases as well. This also means that the computational cost increases with the number of cells, and the method is not scalable for large values of N. Ultimately, if we want to model floods or tsunamis for the real world, we need a scalable method. This is also some of the motivation for using data-driven methods, as we will investigate if they can be more scalable.

4.2 Data-driven results

In this section we present the results of the data-driven models. We go through two main cases. The first case is the 1D shallow water equations with Gaussian initial conditions, where we compare the performance of the CNN and FNO models. The second case is the 1D linearized shallow water equations in spherical coordinates, where we compare the performance of the CNN and FNO models. We also compare the performance of the models for different values of σ, i.e., the standard deviation of the Gaussian function. We do this to see how the models perform for different types of initial conditions, depending how smooth or discontinuous the solution is.

4.2.1 1D SWE with Gaussian initial conditions

We start by showing the numerical solution of the shallow water equations, then we present the predictions of the NN and FNO models. Until now, we have mostly considered discountinuous initial conditions, but we will also consider smooth initial conditions in this section. We solve the SWE with the following initial conditions:

$$\begin{aligned} h(x,0) &= h_0 \exp\left(\frac{-(x-\mu)^2}{2\sigma^2}\right), \\ u(x,0) &= 0, \end{aligned} \tag{4.2.1}$$

where $h_0 = 1, \mu = 0.5, \sigma = 0.1$. The function $h(x,0)$ is a Gaussian function, illustrated in Figure 4.14.

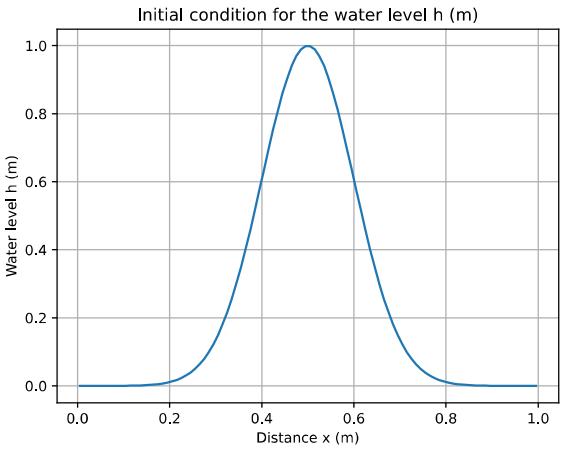

Figure 4.14: Initial conditions (4.2.1).

The domain is $x \in [0, 1]$ with $N = 200$ points and the final time is $t = 1.0$. We use a CFL number of 0.9 and variable time steps. The numerical solution is shown in Figure 4.15, in both a contour plot and a 3D plot.

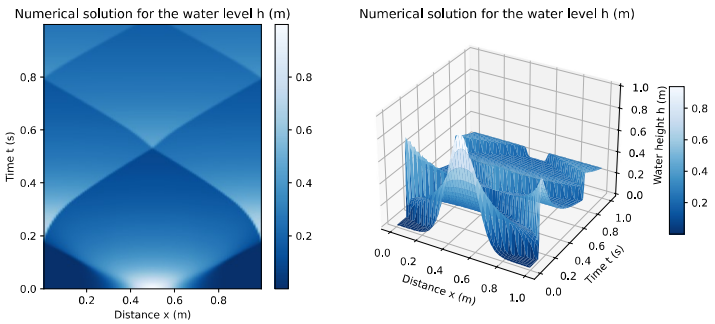

Figure 4.15: Numerical solution of the shallow water equations with initial conditions (4.2.1).

CNN Model

In the convolutional neural network, we train the model using the data generated by the numerical solution of the shallow water equations. The model uses the data from the numerical solution to predict the solution at the next time step. Meaning that the input and output data are the same, but shifted one time step. This way, the model is supposed to learn the flowmap. The model has been trained using the Adam optimizer with a learning rate of 0.001, a batch size of 32.

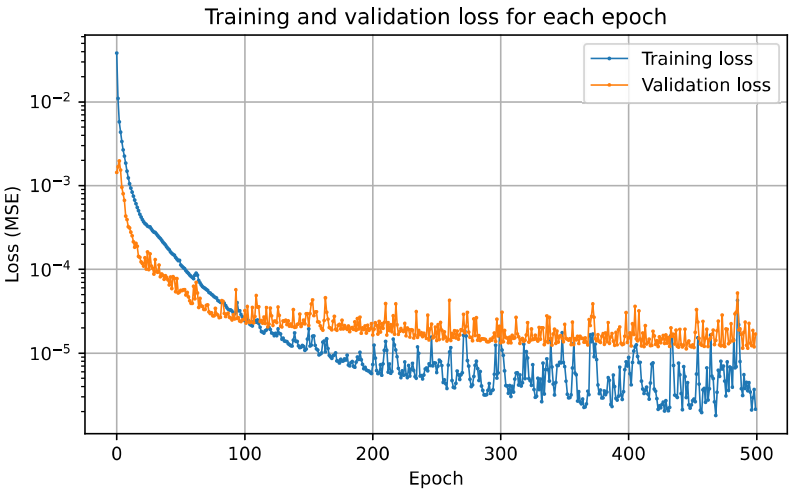

Figure 4.16: Training and validation loss for the CNN model.

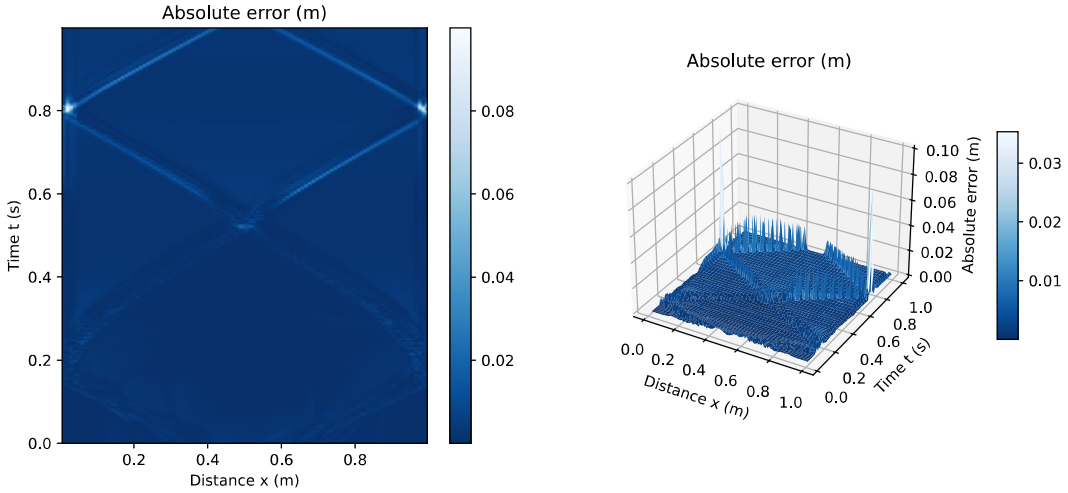

Figure 4.17: Error plot for the predictions for the CNN model.

To understand the performance of the model, we consider the predictions for some given time steps, shown in Figure 4.18.

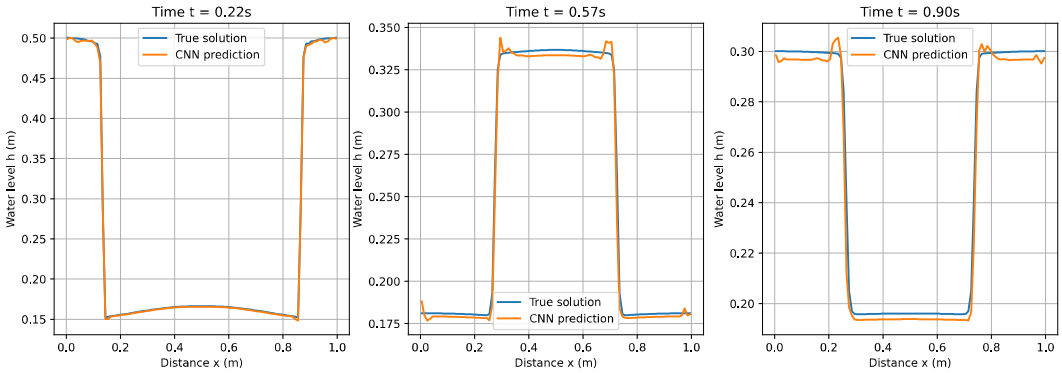

Figure 4.18: Predictions for the CNN model for some given time steps.

We also see that the highest absolute errors are located at the edges of the solution, which is expected, as the solution tends to be discontinuous. We also see that the further we get in time, the worse the predictions become. This is also somehow expected, since we are outside the training data.

Based on the training and validation loss, the model is overfitting the data. We see that as the training loss is decreasing, the validation loss is increasing. This also means that the model is not able to generalize the data. A possible solution could be to use more data.

Again, we see that the model finds some of the dynamics, but has many oscillations. We have also trained a FNN model and a LSTM model, but the results are not shown here, as the performance is worse than the RNN model.

FNO Model

One of the main goals in this thesis is to use Fourier Neural Operators to solve the shallow water equations. We define a FNO model, which consists of an input channel, 64 hidden channels and an output channel. We use a Fourier basis with 16 modes and a batch size of 32. The model is trained using the Adam optimizer with a learning rate of 0.001, a total of 1000 epochs and the critera is to minimize the mean squared error (MSE). The model is trained on the same data as the CNN model, and tested on the same data.

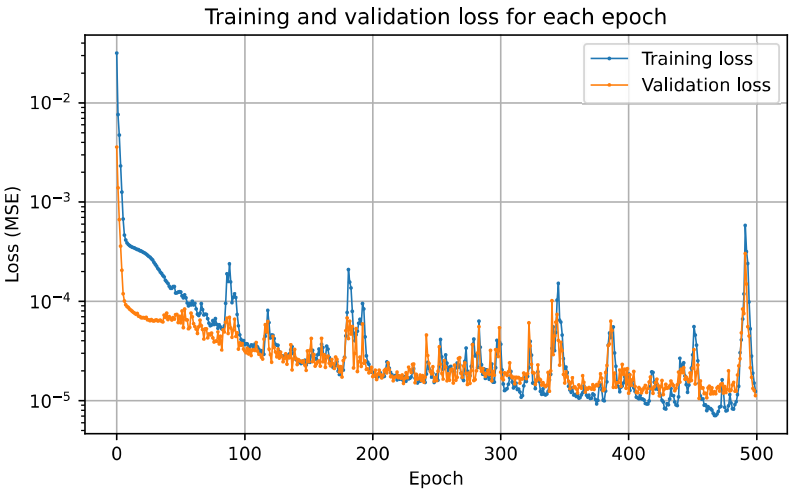

Figure 4.19: Training and validation loss for the FNO model.

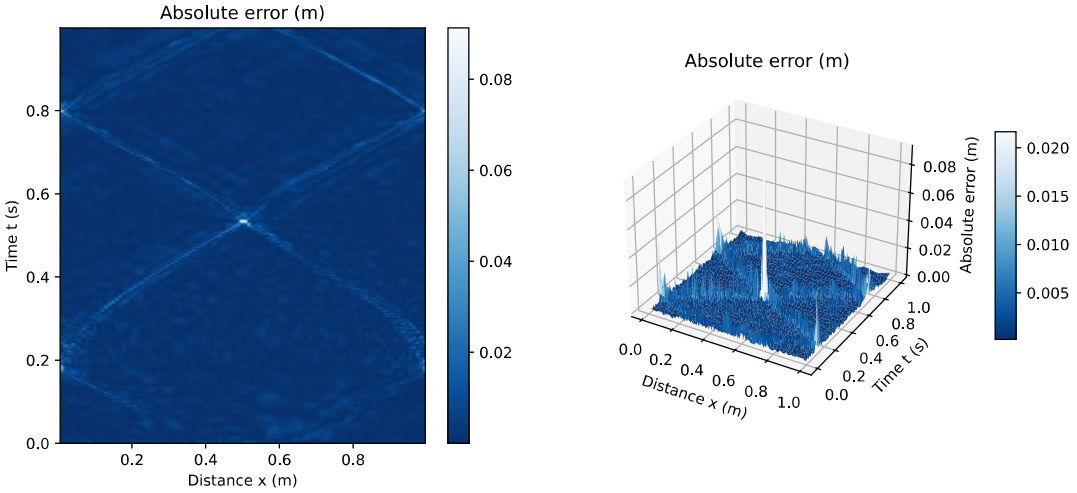

Figure 4.20: Error plot for the predictions for the FNO model.

To get an overview of the performance of the model, we consider the predictions for some given time steps, shown in Figure 4.21.

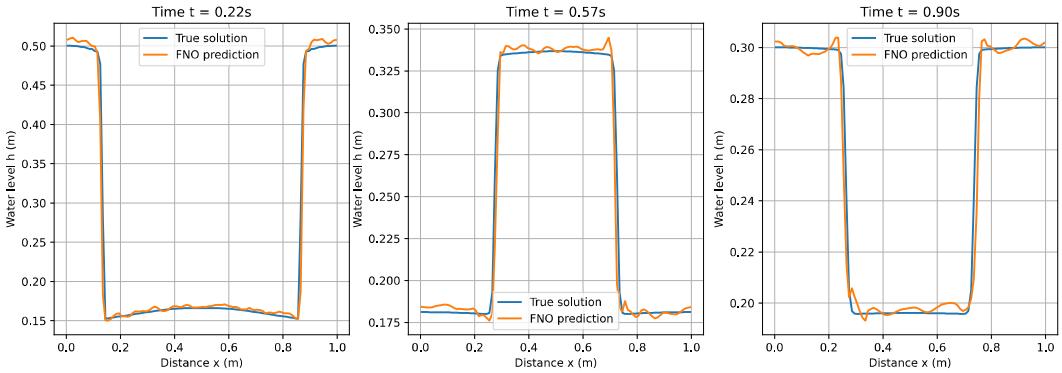

Figure 4.21: Predictions for the FNO model for some given time steps.

Comparison

UDFYLDT MED 1D CNN Gauss + 1D FNO Gauss (ikke for Toro).

Model	Gauss initial condition				Toro test case			
	Epochs	MSE	MAE	Time (s)	Epochs	MSE	MAE	Time (s)
CNN	500	2.25e-05	3.30e-03	52.63	100	5.80e-04	1.43e-02	46.73
FNO	500	1.04e-05	2.10e-03	223.11	100	1.33e-04	6.03e-03	372.20

Table 4.4: Test loss in terms of MSE and MAE, and time for training the models for the 2D SWE.

4.2.1.1 FNO Toro test 1

To test the FNO model on a more challenging problem, we consider the Toro test case 1. We use the same model as before, but we train it on the data from the Toro test case 1. Again, we use the first 80% of the data for training and the last 20% for testing. The results of the model are shown in Figure 4.23 and **??**.

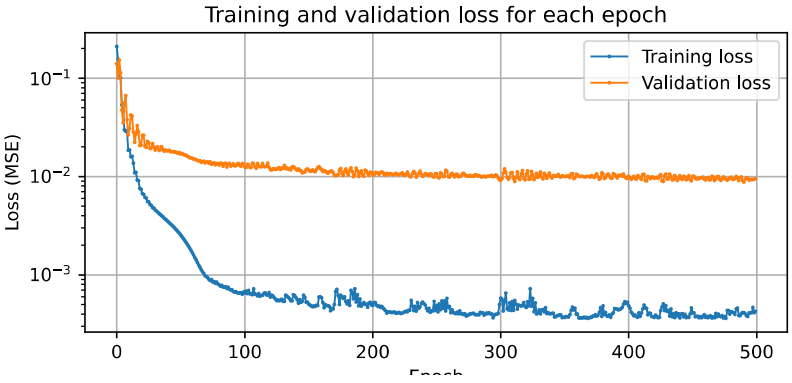

Figure 4.22: FNO Toro test 1 loss.

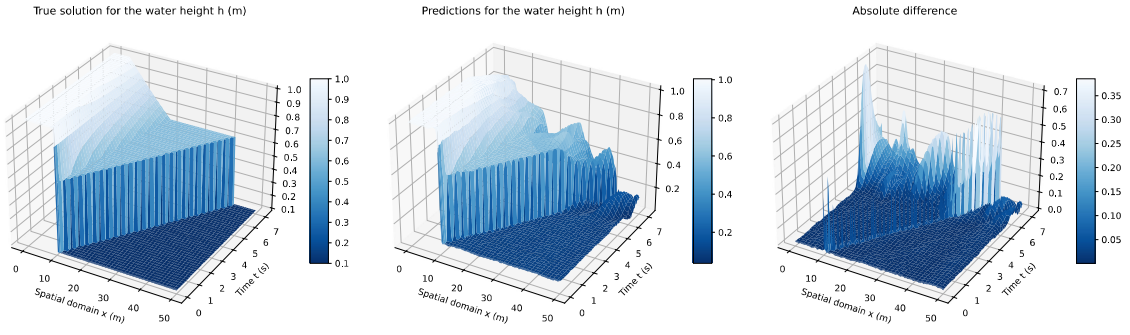

Figure 4.23: FNO Toro test 1 predictions in 3d.

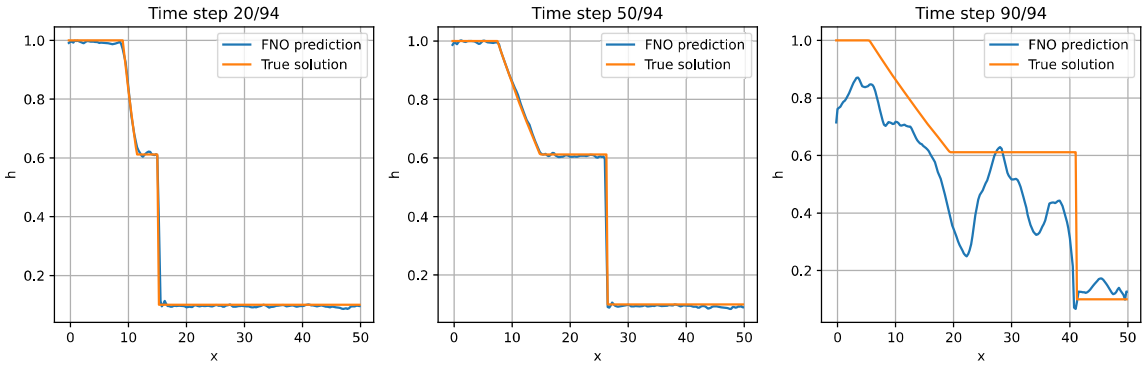

Figure 4.24: FNO Toro test 1 predictions timesteps.

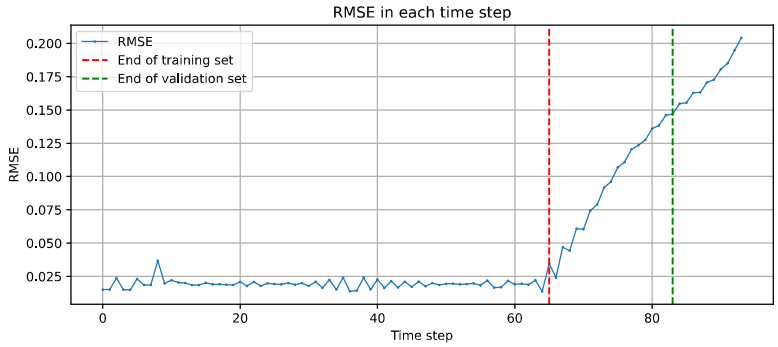

Figure 4.25: FNO Toro test 1 predictions RMSE.

4.2.2 1D linearized SWE in Spherical Coordinates

We also consider the spherical shallow water equations in a 1D setting, focusing on the linearized SWE on a circular domain. The length of the domain corresponds to the circumreference of the circle, $L = 2\pi$, and is discretized into

$N = 500$ points. The initial conditions is specified as a Gaussian function wrapped around the circle, expressed as:

$$h(\theta, 0) = h_0 \exp\left(\frac{-(\theta - \mu)^2}{2\sigma^2}\right),$$

where the parameters are $h_0 = 1, \mu = \frac{\pi}{4}, \sigma = \frac{\pi}{16}$. The initial conditions can be seen in Figure 4.26.

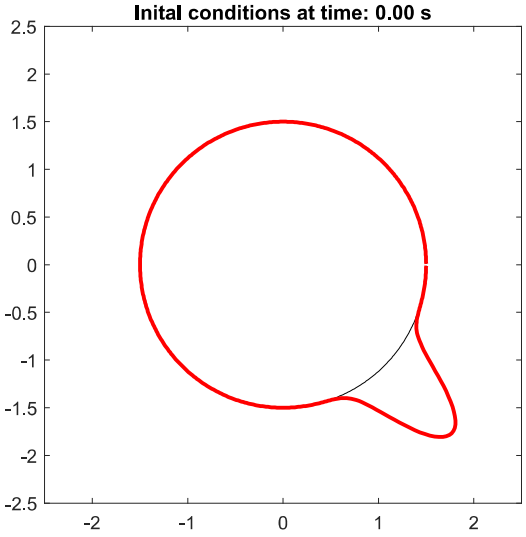

Figure 4.26: Initial conditions for the 1D linearized shallow water equations in spherical coordinates.

The numerical solution in the θ, t plane is shown in Figure 4.27, for $t = 0$ to $t = 1$.

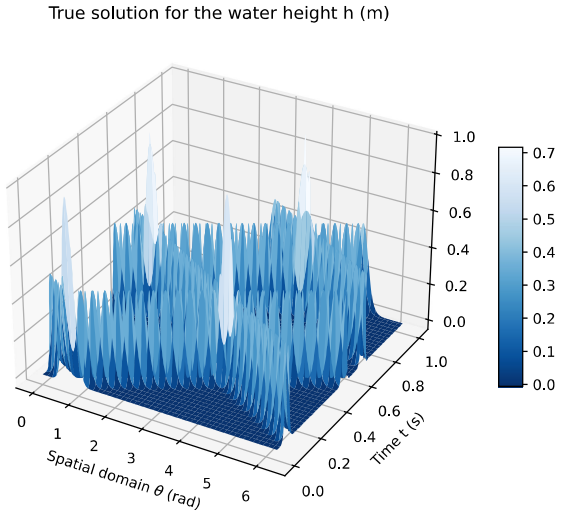

Figure 4.27: Numerical solution of the spherical shallow water equations in 1D in the $\theta, t-$space.

Based on the data generated by the numerical solution (we use time steps of $\Delta t = 0.0025$), we train a FNO model and a Convolutional Neural Network (CNN) model.

CNN Model

We have also trained a CNN model on the data generated by the numerical solution of the SWE in 1D. Like the FNO model, the CNN model is trained on the data from $t = 0$ to $t = 0.6$, validated on the data from $t = 0.6$ to $t = 0.8$, and tested on the data from $t = 0.8$ to $t = 1.0$. The training and validation loss is shown in Figure 4.28.

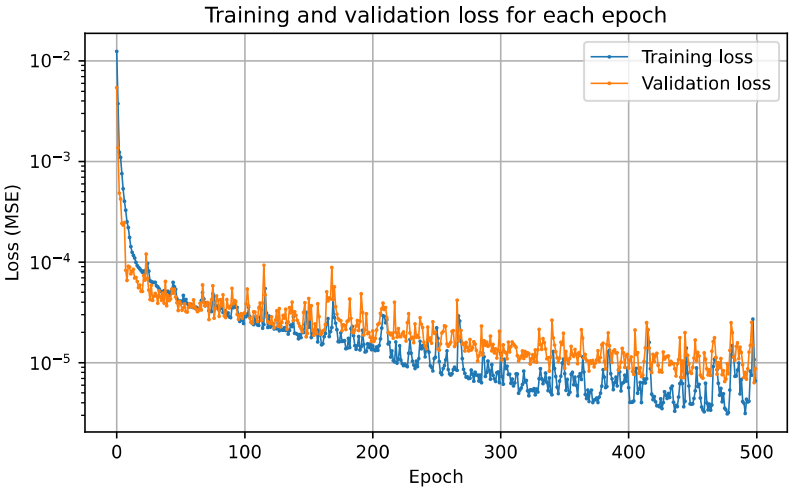

Figure 4.28: Training and validation loss for the CNN model for the spherical shallow water equations in 1D.

The error plots are shown in Figure 4.29.

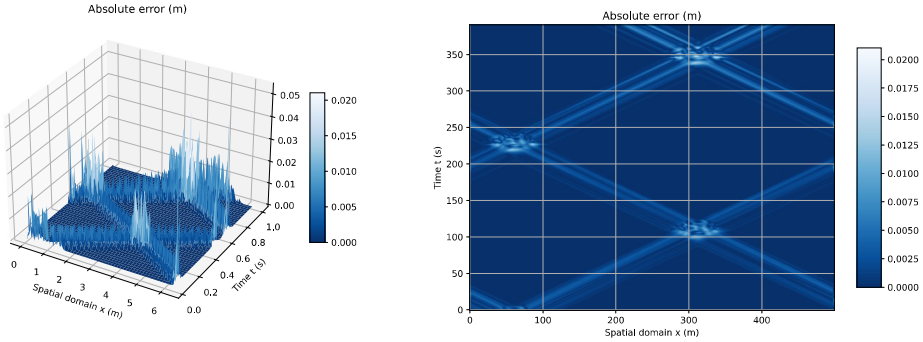

Figure 4.29: Error plots for the predictions of the CNN model for solving the 1D linearized spherical SWE.

The predictions for some given time steps are shown in Figure 4.30.

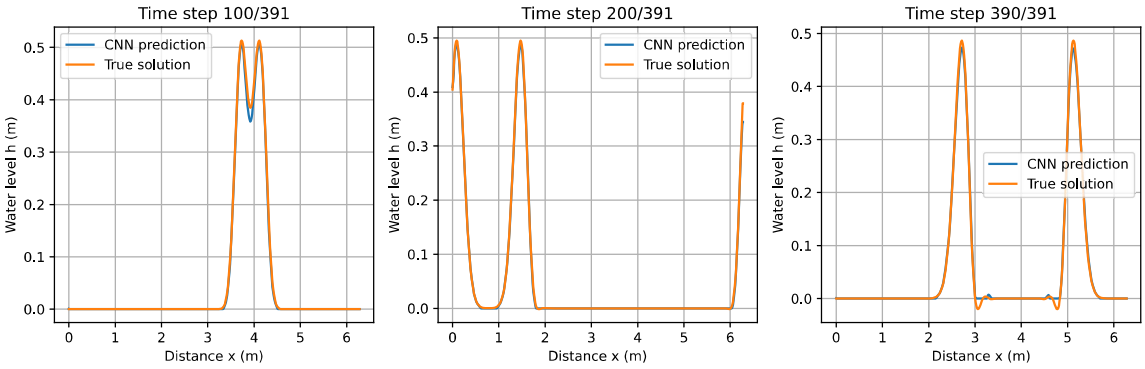

Figure 4.30: Predictions for the spherical shallow water equations in 1D using the CNN model for some given time steps.

From Figure 4.30, we also see that the predictions capture the waves, but are more noisy than the FNO predictions.

FNO model

The FNO model consists of an input channel, 64 hidden channels and an output channel. We use a Fourier basis with 16 modes and a batch size of 32. The model is trained using the Adam optimizer with a learning rate of 0.001, a total of 100 epochs and the critera is to minimize the mean squared error (MSE). The model is trained on the data from $t = 0$ to $t = 0.6$, validated on the data from $t = 0.6$ to $t = 0.8$, and tested on the data from $t = 0.8$ to $t = 1.0$. The training and validation loss is shown in Figure 4.31.

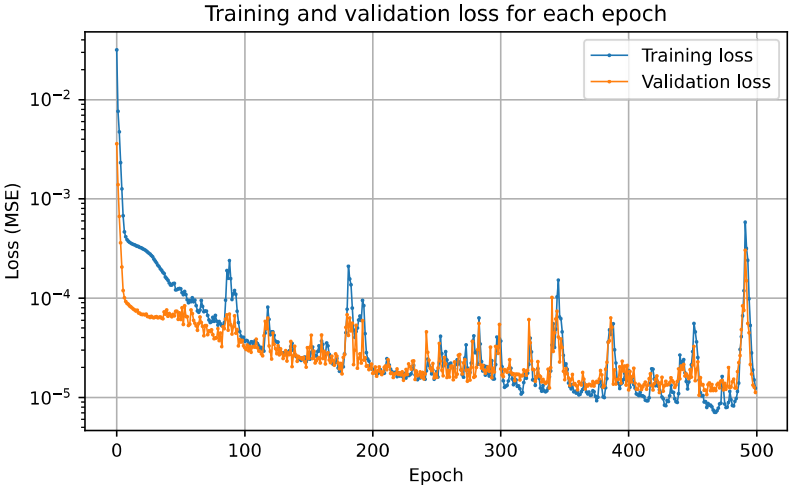

Figure 4.31: Training and validation loss for the FNO model for the spherical shallow water equations in 1D.

The error plots are shown in Figure 4.32.

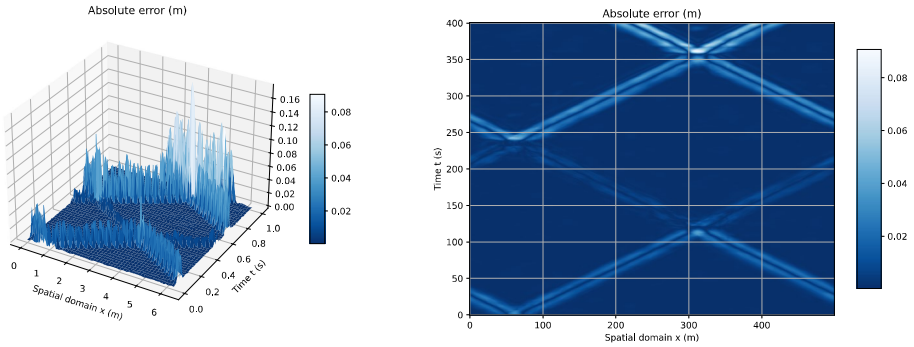

Figure 4.32: Error plots for the predictions of the 1D linearized spherical SWE.

From Figure 4.32 we see that the model is able to learn the dynamics of the solution. We also see that the absolute error is biggest at the edges of the solution, which is expected, as the solution tends to be discontinuous.

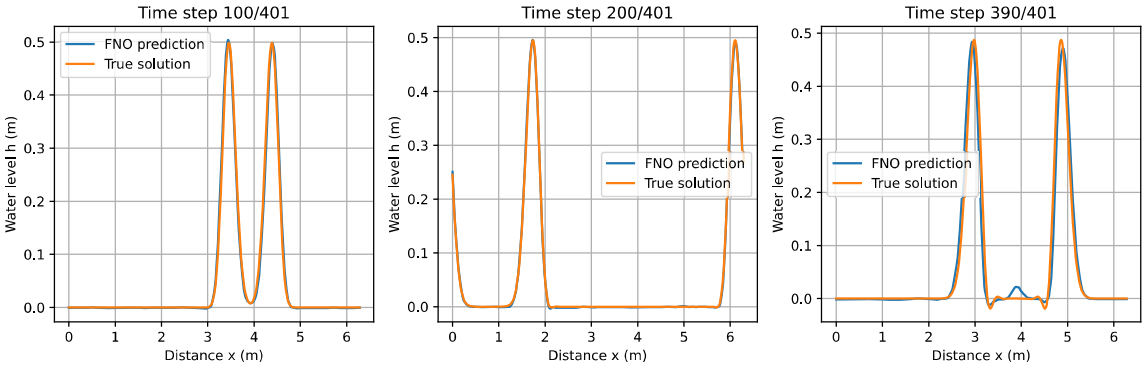

Figure 4.33: Predictions for the spherical shallow water equations in 1D.

From Figure 4.33 we see that the FNO predictions are smooth and rather accurate, but have some small errors in the top of the waves.

Comparison

To get an overview of the performance of the different models, we consider the MSE for the predictions for the 1D SWE spherical case.

Model	$\sigma = \pi/8$			$\sigma = \pi/16$			$\sigma = \pi/32$		
	MSE	MAE	Time (s)	MSE	MAE	Time (s)	MSE	MAE	Time (s)
CNN	1.31e-02	3.34e-02	91.33	2.32e-05	1.95e-03	112.70	5.92e-03	2.85e-02	8.48
FNO	2.05e-04	8.23e-03	108.79	5.78e-04	1.04e-02	564.82	5.19e-04	1.16e-02	125.80

Table 4.5: Test loss in terms of MSE and MAE, and time for training the models for the 1D spherical SWE.

From Table 4.5 we see that the CNN model is slightly faster and better than the FNO model for $\sigma = \pi/8$ and

$\sigma = \pi/16$, but for $\sigma = \pi/32$, the performance of the CNN model is decreasing. Probably due to the fact that the smaller the σ, the more discontinuous the solution is, and the FNO model is better at capturing the discontinuities. We see that the MAE in general is higher than the MSE, and is also increasing for smaller σ. Additionally, we observe that the MAE is higher, as it places more weight on small errors compared to the MSE. And in the CNN case we see a lot of small errors/noise, which is why the MAE is higher than the MSE.

4.3 Data-driven results for the 2D SWE

In this section we will present the results for the 2D SWE using the data-driven models. We consider the same initial condition (Gaussian function) as in the 1D case, but now in two dimensions. We solve the 2d SWE using both a CNN and a FNO model, and compare the results in terms of run time and accuracy. We also compare the run time to the numerical method FVM, to see if the data-driven models can be used as a faster alternative to the FVM. We also test the ability of the FNO model to generalize to a finer grid, by training the model on a coarse grid and then making predictions on a fine grid. Finally, we will test the FNO model's ability to generalize further in time and make long-term predictions.

4.3.1 2D SWE with initial Gaussian function

The initial condition for the 2D problem is a Gaussian function with a standard deviation of 0.1 and a mean of 0.5. The initial condition can be seen in Figure 4.34.

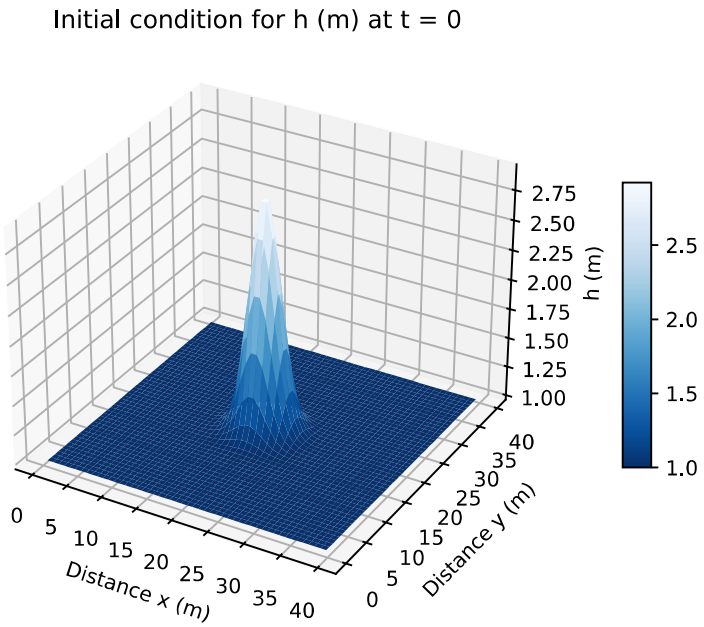

Figure 4.34: Initial condition for the 2D problem.

CNN Model

The training and validation loss for the 2D CNN model can be seen in Figure 4.35.

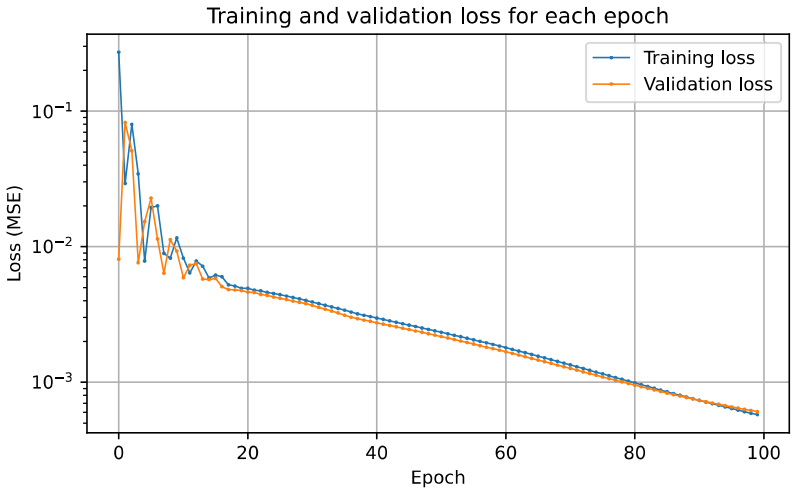

Figure 4.35: Training and validation loss for the 2D CNN model.

The error plot for the last prediction for the 2D CNN can be seen in Figure 4.36.

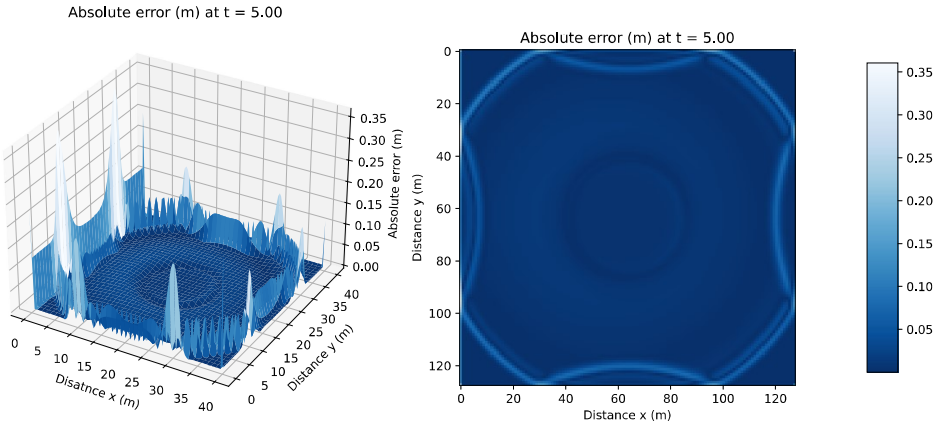

Figure 4.36: Error plot for the last prediction for the 2D CNN.

FNO Model

The training and validation loss for the 2D FNO model can be seen in Figure 4.37.

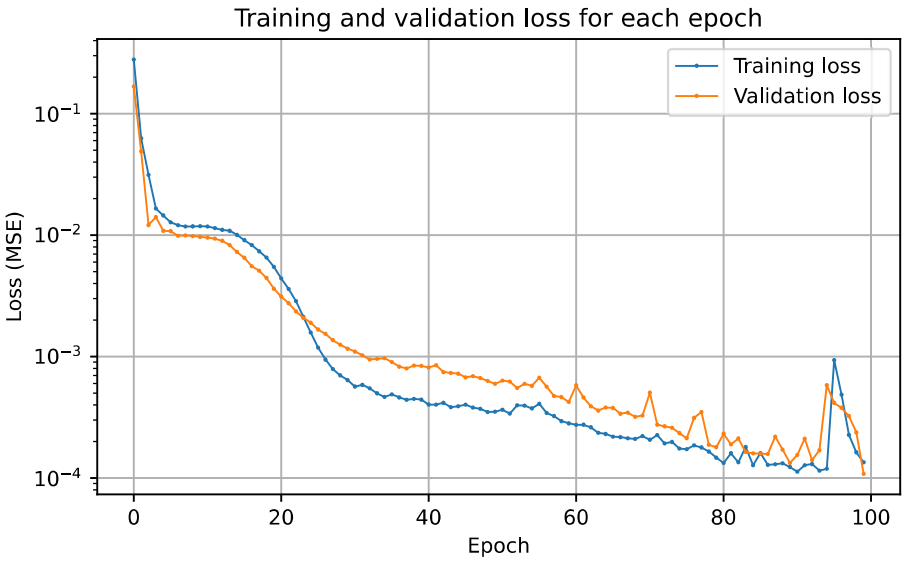

Figure 4.37: Training and validation loss for the 2D FNO model.

The error plot for the last prediction for the 2D FNO can be seen in Figure 4.38.

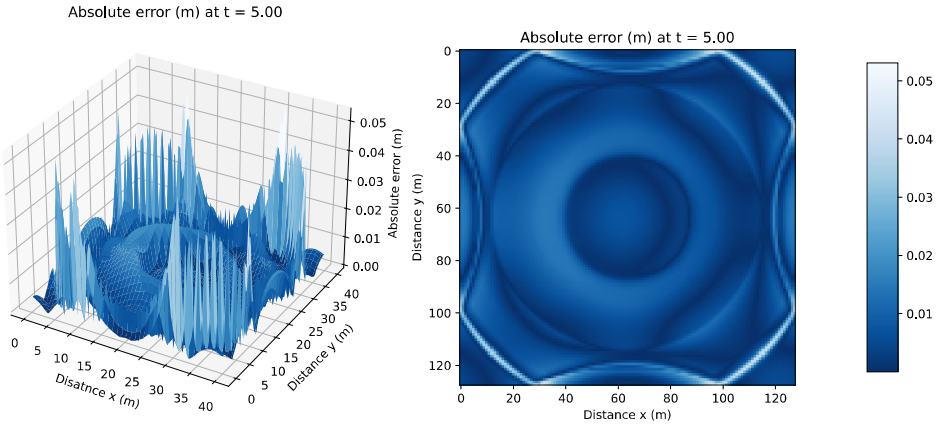

Figure 4.38: Error plot for the last prediction for the 2D FNO.

Comparison

To get an overview of the performance of the different models, we consider the MSE and MAE for the predictions for the 2D SWE case.

Model	$N = 64$				$N = 128$			
	Epochs	MSE	MAE	Time (s)	Epochs	MSE	MAE	Time (s)
CNN	100	4.16e-03	4.13e-02	11.91	100	7.91e-04	1.29e-02	139.59
FNO	100	2.59e-04	8.68e-03	81.11	100	2.33e-04	8.68e-03	780.74

Table 4.6: Test loss in terms of MSE and MAE, and time for training the models for the 2D SWE.

We see that the FNO model in general needs fewer epochs to converge compared to the CNN model, but it also takes longer time to train. From the theory we know that FNO are supposed to work when we are training on a coarse grid and then make predictions on a fine grid. To test this, we will train the models on a coarse grid and then make predictions on a fine grid. The table below shows the results when the FNO-model is trained on a grid with $N = 64$ and then makes predictions on a grid with $N = 128$.

Model	$N = 128$		
	MSE	MAE	Prediction time (s)
FNO	1.07e-04	6.34e-03	7.00e+00

Table 4.7: Test loss in terms of MSE and MAE, and time for training the FNO model on a grid with $N = 64$ and then making predictions on a grid with $N = 128$.

From the results in Table 4.7 we see that the FNO model is able to generalize to a finer grid. This way, it is possible to train the model on a coarse grid and then make predictions on a fine grid, which is a great advantage when solving the SWE numerically, as it is very computationally expensive to solve the SWE on a fine grid using the FVM.

We will also time the predictions to compare the speed of the models. We can also compare to the run time of the numerical method FVM, see Table 4.3. Handle the scalability isssues.

Model	$N = 256$				
	Epochs	MSE	MAE	Training time (s)	Prediction time (s)
CNN					
FNO					

Table 4.8: Test loss in terms of MSE and MAE, and time for training the models for the 2D SWE.

Long-term predictions

We are also very interested in testing the ability of the FNO to generalize further in time. We will therefore train the model on a time interval $[0, 5]$ and make predictions for $t = X$. Our first approach was to make a prediction, and based on that prediction make a new prediction for the next time step. This way we can make predictions for a longer time interval. For the first time steps the results were fine, but as we made predictions further in time, the error increased, probably because the error from the previous time step was carried over to the next time step, i.e., the error was accumulated. So, the next approach was to make sequences of the data and then train the model to predict the next time step based on the sequence. This way, the newest prediction becomes a part of a sequence and thus, do not have the same influence on the next prediction as in the previous approach. The hope is that this approach makes long-term predictions more stable. We make predictions for up to $n = 20$ time steps in the future. The results for the long-term predictions can be seen in Figure 4.39.

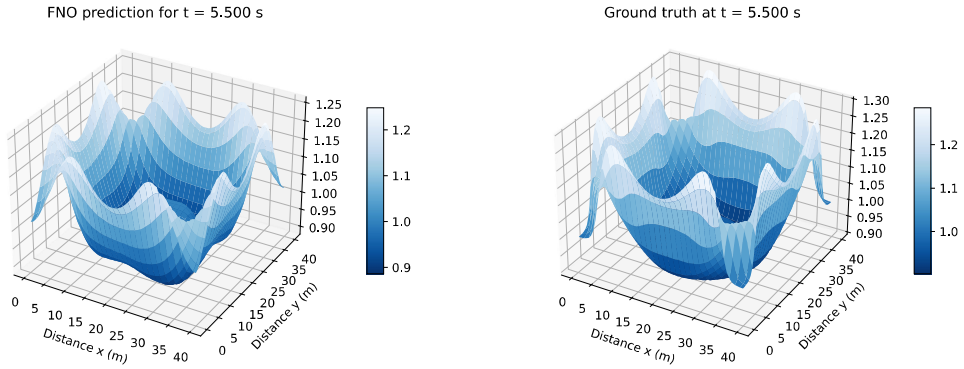

Figure 4.39: Long-term prediction for the 2D FNO model.

The error plot for the long-term predictions can be seen in Figure 4.40.

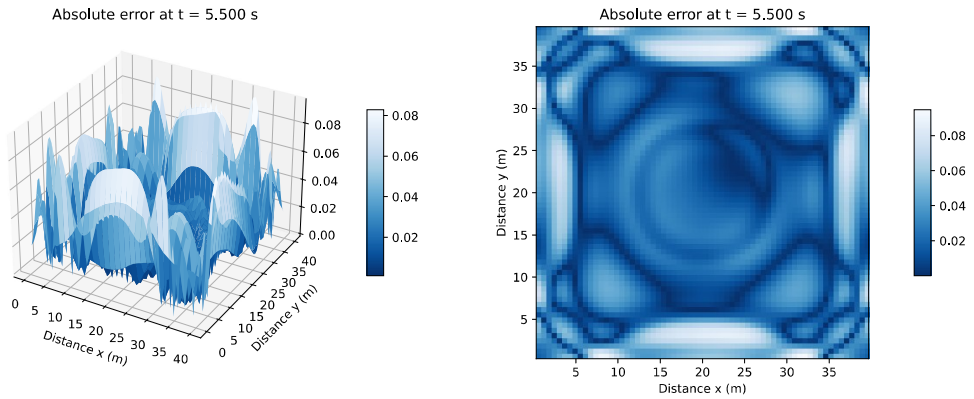

Figure 4.40: Error plot for the long-term prediction for the 2D FNO model.

From the error plot in Figure 4.40, we see that the error is only slightly bigger than for the short-term predictions in Figure 4.38. This indicates that the FNO model is able to generalize further in time and make long-term predictions.