Physics-Based Tsunami Modeling for Machine Learning Applications

By

DAVID PATRICK GRZAN
DISSERTATION

Submitted in partial satisfaction of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

Physics

in the

OFFICE OF GRADUATE STUDIES

of the

UNIVERSITY OF CALIFORNIA

DAVIS

Approved:

Chair John B. Rundle, Chair

David Wittman

Shirley Chiang

Committee in Charge

2022

Contents

ABSTRACT

Tsunamis in the last two decades have resulted in the loss of life of over 200,000 people and have caused billions of dollars in damage. Therefore there is great motivation for the development and improvement of current tsunami warning systems. However, this is quite difficult as tsunamigenic sources such as earthquakes are inherently unpredictable due to their non-linear and chaotic nature. As a result, forecasting methods focus on taking advantage of the short span of time between the earthquake rupture and the wave reaching the shore. A common method includes forward modeling, which first obtains the earthquake source by using inversion techniques. It is then used as the initial condition for the tsunami and is simulated in real time to obtain a forecast of the coast. However, methods such as these still take an appreciable amount of time to run and rely on supercomputers which may not be available to countries or institutions which do not have sufficient resources and infrastructure. Therefore, in response to the shortcomings of current methods, other methods have been explored utilizing neural networks trained on precomputed tsunami databases to make inundation forecasts. From the nature of neural networks, these methods obviate the need for supercomputers and cut down the forecasting computation time significantly.

The work presented in this dissertation represents advancements made towards the creation of a neural network-based tsunami warning system which can produce faster inundation forecasts with increased accuracy. This was done by first improving the waveform resolution and accuracy of Tsunami Squares, an efficient cellular automata approach to wave simulation. It was then used to create a database of precomputed tsunamis in the event of a magnitude 9+ rupture of the Cascadia Subduction Zone, located only $\sim$100 km off the coast of Oregon, US. Two methods were tested in the effort to link readily available sensor data directly to inundation forecasts. One approach utilized a convolutional neural network which took wave height data from buoys as input and proved successful as maps of maximum inundation could be predicted for the town of Seaside, Oregon with a median error of $\sim$0.5 m.

ACKNOWLEDGMENTS

Research by David Grzan was supported under NASA grant NNX12AM22G, and by DoE grant DE- SC0017324 to the University of California, Davis. Portions of this research were also carried out at the Jet Propulsion Laboratory, California Institute of Technology under contract with NASA. Additional thank you to Tony Song for providing the ROMS simulation data used for comparison. ROMS is available from https://www.myroms.org.

Chapter 1

Introduction

1.1 Motivation

Tsunamis in the last two decades have caused the loss of over hundreds of thousands lives and have resulted in billions of dollars of damage. The particularly destructive tsunamis mentioned are caused by megathrust earthquakes along the Indo-Pacific convergent plate boundaries also known as The Ring of Fire. The 2004 Mw 9.2 Sumatra-Andaman Earthquake and Great Indian Ocean tsunami (Ammon et al., 2005; Ishii et al., 2005; Lay et al., 2005; Stein & Okal, 2005; Subarya et al., 2006) resulted in over 230,000 casualties. The greatest loss of life occurred along the Sumatra coastline nearest the earthquake epicenter from tsunami inundation heights of up to 30 m (Paris et al., 2009). The Mw 8.8 2010 Maule earthquake in Chile (Lay et al., 2010; Delouis et al., 2010) resulted in 124 tsunami related fatalities and wave heights up to 15-30 m along the coast nearest the epicenter (Fritz et al., 2011). The 2011 Mw 9.0 Tohoku-oki earthquake in Japan (Simons et al., 2011; Lay et al., 2011) generated a tsunami with inundation amplitudes as high as 40 m and left over 15,000 casualties (Mori & Takahashi, 2012). Although work has been done to improve existing warning systems, more sophistication and availability is needed to prevent future disasters such as these.

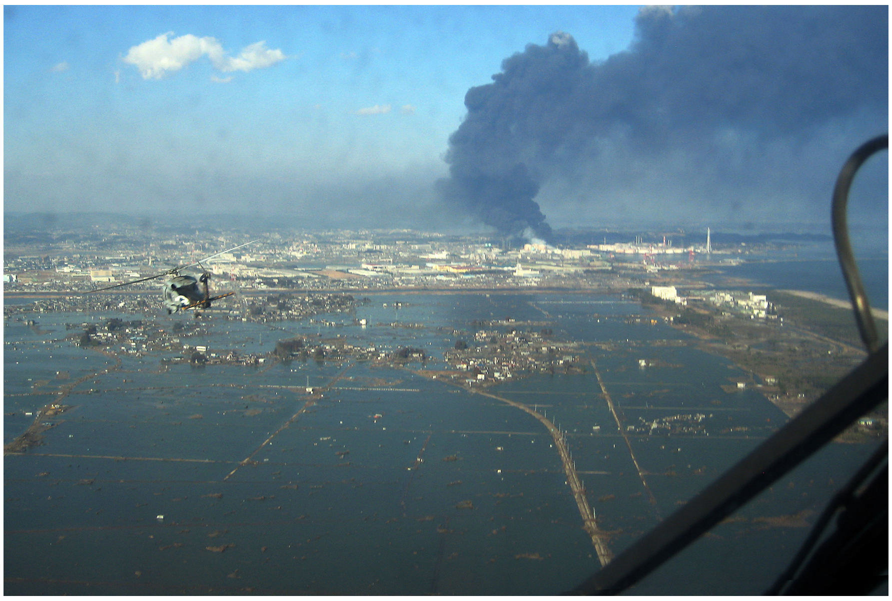


Figure 1.1: Aftermath of the 2011 Tohoku, Japan event in the Miyagi Prefecture.

1.2 Tsunami Sources

Tsunamis can be generated from a number of different sources including underwater earthquakes, landslides, volcanic eruptions, and asteroids. In this dissertation, tsunamis generated from submarine earthquakes will be focused on as they are the most deadly and best understood. Typically, tsunamigenic earthquakes are those which are underwater, commonly occurring along major subduction zone plate boundaries in the Pacific Ocean. Subduction zones belong to a class of faults known as reverse-faults in which a heavier tectonic plate dives under a lighter plate at a shallow angle generating stress over time as the plates move past each other (Stern, 2002). After decades or even centuries of stress buildup, the plates slip, causing uplift of the seafloor over hundreds of kilometers. This uplift of the seafloor in turn lifts the water above it as well, causing what is known as a tsunami.

When a fault ruptures, the rock around it deforms in a particular pattern. For explanation purposes, the fault can be described as a perpendicular cut into the Earth's surface. During the rupture, the two sides of the cut move past each other, keeping the corners stationary. This produces the deformation pattern shown in Figure 1.2.

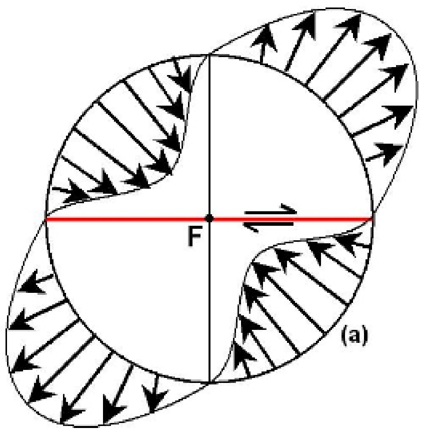


Figure 1.2: The deformation pattern of the surrounding ground for a fault where the top side of the fault is slipping the to the right.

In tsunamigenic earthquakes, a slip between two underwater bodies of rock causes a three dimensional deformation of the seafloor, where the vertical component is responsible for the displacement of the water column above. The resulting seafloor displacement ultimately determines the height of the wave and depends on the mean vertical slip of the seafloor and on the area of displacement (Röbke & Vött, 2017).

Applying these principles to the 2011 Tohoku Earthquake yields the three dimensional surface displacement shown in Figure 1.3.

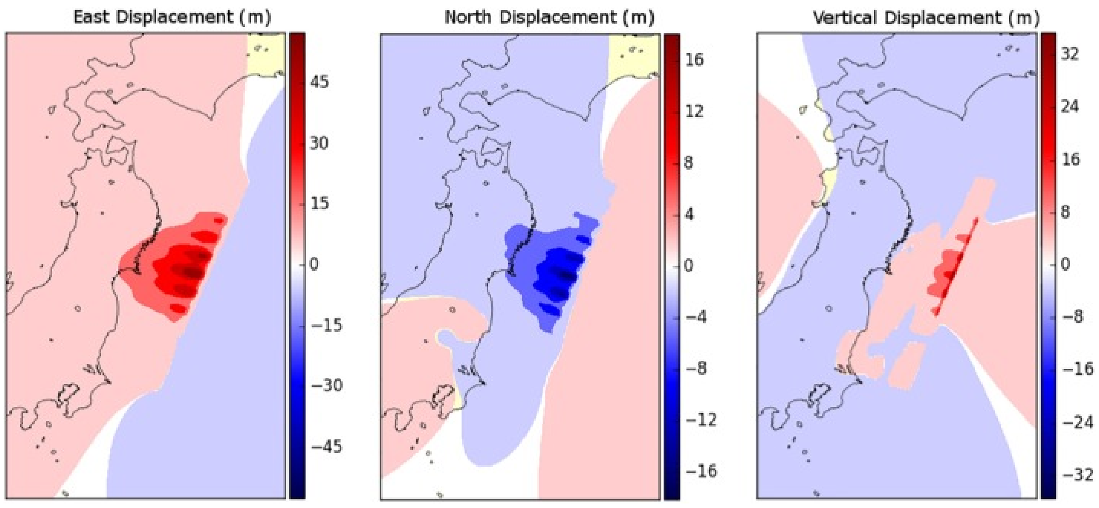


Figure 1.3: Simulated deformation of the seafloor for all three components of the displacement caused by the 2011 Tohoku Earthquake. (Wilson et al., 2020)

Since the duration of a large earthquake typically ruptures over a few minutes (such as the 2011 Tohoku Earthquake which lasted 2.5 min (Satake et al., 2013)), it can be considered short when compared to tsunami travel times on the scale of multiple hours. For this reason, the tsunami simulator used in this dissertation assumes instantaneous uplift for the initial conditions.

1.3 Tsunami Wave Properties

There are three types of ocean waves: shallow water waves, deep water waves, and transition waves. Tsunamis belong to the class of waves defined by having a depth less than 1/20th of their wavelength. Deep water waves are defined by having a depth greater than one half the wavelength, whereas transition waves fall in between these constraints.

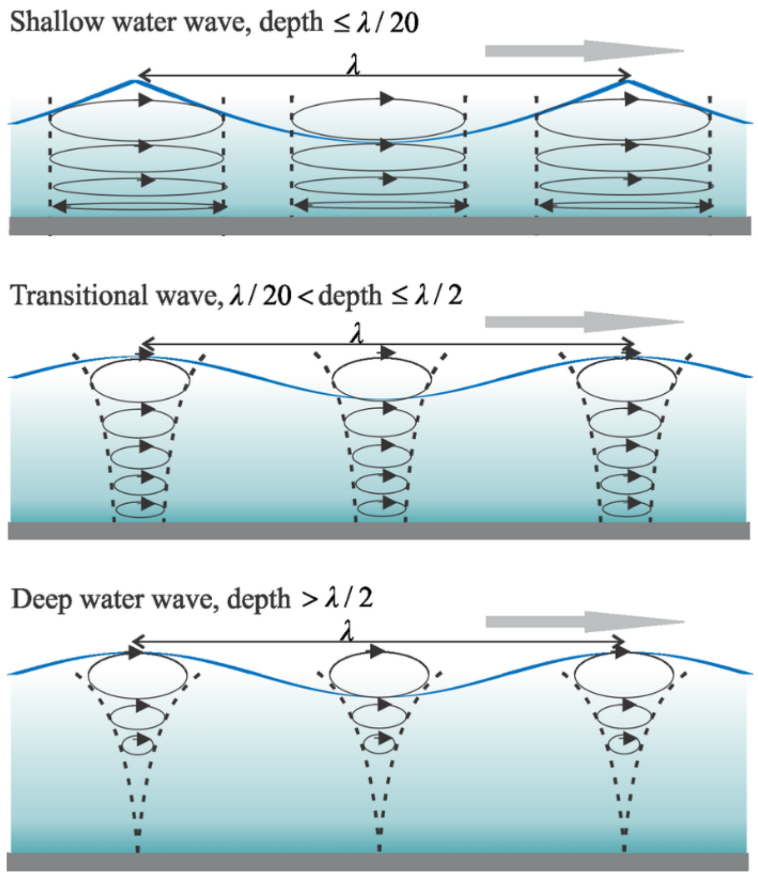

Shallow water wave, depth $\leq \lambda / 20$
λ
Transitional wave, $\lambda / 20 < \text{depth} \leq \lambda / 2$
λ
Deep water wave, depth $> \lambda / 2$
λ

Figure 1.4: Definitions of shallow water, deep water, and transition waves.

Tsunami waves are classified as shallow water waves with particles exhibiting the orbital patterns seen in the top image of figure 1.4. This orbital motion is what transmits energy in a water wave. Unlike common examples of longitudinal and transverse waves such as acoustic waves or vibrational waves along a string, the restoring force for a water wave is gravity. Although the wavelike properties between all three examples are similar, the mechanism by which water waves are propagated is completely different, giving rise to unique characteristics.

To give better intuition as to how surface gravity waves propagate, consider an initial fluctuation up from the surface of a body of water. This fluctuation is then pulled down by gravity towards equilibrium and in doing so gains inertia which penetrates down below the surface. The fluctuation then propagates horizontally because any difference of height on the surface of the water causes a pressure gradient which accelerates the fluid. When the water is accelerated, it piles up ahead, thus moving the disturbance a certain distance. The net effect is a traveling wave.

The wavespeed of a water wave can be derived by obtaining the wave equation starting from the Navier-Stokes equations and applying boundary conditions along with the necessary assumptions. Sparing the derivation the wave speed is given in Equation 1.1.

$$v = \sqrt{\frac{g\lambda}{2\pi}\tanh(\frac{2\pi d}{\lambda})} \tag{1.1}$$

By taking the shallow water limit where $\lambda \gg d$, $tanh(\frac{2\pi d}{\lambda})$ becomes $\frac{2\pi d}{\lambda}$ and we obtain the familiar equation for the shallow water wavespeed in Equation 1.2

$$v = \sqrt{gd} \tag{1.2}$$

The shallow water wavespeed is a peculiar result because it depends only on the depth, not on the properties of the water itself. To contrast, deep ocean waves behave in a way where the wave speed is wavelength-dependent, giving rise to a dispersion relation. The relationship between the wavespeed and wavelength for deep water waves is given in Equation 1.3.

$$v = \sqrt{\frac{g\lambda}{2\pi}} \tag{1.3}$$

This equation is obtained by starting with Equation 1.1 and applying the deep water criteria, $\lambda \ll d$. In doing so, tanh is approximately 1. This is simply the limit of tanh as the argument approaches infinity.

Intuitively, this type of wave is best explained by comparison to a pendulum since they share the same restoring force, gravity. Both a vibrating string wave and a pressure wave have no obvious comparisons to a deep water wave since there is no wavelength dependence for these types of waves. Essentially, one can imagine the pendulum length being analogous to the wave's wavelength. When the length of a pendulum or the wavelength of a deep water wave is increased, both their angular velocities are decreased. Equations 1.4 and 1.5 show the similarity.

$$\omega_{deepwater} = \sqrt{\frac{2\pi g}{\lambda}} \tag{1.4}$$

$$\omega_{pendulum} = \sqrt{\frac{g}{L}} \tag{1.5}$$

However, the shallow water wave speed $\sqrt{gd}$ has a different intuition and must be treated separately from deep water waves. Referring to figure 1.4, the particle paths for a shallow water wave are nearly horizontal compared to the nearly circular deep water particle paths. It is also important to note that the shallow water particle paths do not change in width as they penetrate down to the sea floor. The important distinction between shallow and deep water waves is that deep water particle paths decay before reaching the seafloor while shallow water particle paths are practically unchanging from the surface to the sea floor. This implies two things. (1) Since deep water particle paths decay before reaching the bottom, this means deep water waves cannot be depth dependent. The depth could be infinite and it would not change the nature of the wave. (2) For a shallow water wave, this implies that the waves "feel" the bottom since the particle paths reach the bottom. In other words, if the depth were shallower it would limit the amount of water motion and if the depth were deeper it would allow more water to move. Therefore, shallow water waves have to depend on depth.

This intuition seemed to be the prevailing answer given to me by the professors I asked, but to me it was still was not very satisfying. I kept looking and found another way of thinking about it by comparing them to acoustic waves.

Looking back to acoustic waves, I described how the wavespeed depended on the quantity $\frac{dP}{d\rho}$. In short, this quantity is defined as the "amount of push back the particles feel when squeezed together". It is also reasonable to assume this is proportional to the particle density of the substance (not density, just particle density). The intuition gained from these acoustic waves can be applied to shallow water waves if we replace the particle density with depth. Just as the depth is proportional to the amount of material in a shallow water wave, the particle density is proportional to the amount of material in an acoustic wave. If we increase the particle density of a medium, an acoustic wave will travel faster. Similarly if we increase the depth of water, a shallow water wave will travel faster. This comparison is made possible because of the strictly horizontal motion of the water as a wave is propagated. It parallels the strictly one dimensional, back and forth motion of particles which propagate acoustic waves.

1.4 Summary of Results

The work presented in this dissertation represents advancements made towards creating a tsunami warning system which can produce more timely and accurate inundation forecasts. This was done by first improving Tsunami Squares, an efficient cellular automata approach to wave simulation. It was then used to create a precomputed tsunami database on which a neural network was trained to directly generate unique inundation predictions based on available sensor data.

In Chapter 2, the algorithm and method of Tsunami Squares is explained in detail. Chapter 3 then introduces a comprehensive solution to preexisting shortcomings of the simulator. The result is a simulator with significantly higher waveform resolution and accuracy in addition to the implementation of a method to allow the simulator to conserve energy. The improvements position Tsunami Squares to be fit for database creation applications due to the computationally cheap nature of the underlying algorithm and its newfound ability to better distinguish between simulations of similar initial conditions. Chapter 4 then describes how Tsunami Squares is used in the creation of a database of precomputed tsunamis in the event of a magnitude 9+ rupture of the Cascadia Subduction Zone, located only ∼100 km off the coast of Washington and Oregon. Two methods involving neural networks are tested in the effort to link readily available sensor data directly to an inundation prediction. One approach utilizes a convolutional neural network which takes wave height data from buoys as input and proved successful as maps of maximum inundation height could be predicted for the town of Seaside, OR with an average error of ∼0.5 m.

Chapter 2

The Tsunami Squares Simulator

2.1 Introduction

Originally inspired by a landslide simulation technique which simulates individual particles, or “balls” (Ward & Day, 2006), the method was adapted to water particles and named “Tsunami Balls” (Ward & Day, 2008, 2010, 2011). To obviate the need for millions of individual water particles, a new method, “Tsunami Squares”, was created which instead accelerates and transports “squares” of water to be imparted onto a grid, conserving volume and linear momentum (Xiao et al., 2015). Another advantage of Tsunami Squares is that it gives no special treatment for wet and dry cells, allowing the free flow of water from sea to land. This method was tested on the 1982 El Picacho landslide in El Salvador (Wang et al., 2015), the 1792 Unzen-Mayuyama mega-slide in Japan (Wang et al., 2019), and the 2011 Tohoku Tsunami (Wilson et al., 2020). Originally written in Fortran, this method was later ported to C++ and parallelized for reduced computational time (Wilson et al., 2020).

2.2 Typical Tsunami Models

Typical tsunami simulators solve nonlinear continuity and momentum conservation equations to obtain height $H(\mathbf{r}, t)$ and horizontal velocity $\mathbf{v}(\mathbf{r}, t)$ at every specified point for each time step

$$\frac{\partial H(\mathbf{r},t)}{\partial t} = -\nabla\cdot\left[\mathbf{v}(\mathbf{r},t)H(\mathbf{r},t)\right] \tag{2.1}$$

and

$$\begin{aligned}\frac{\partial H(\mathbf{r},t)\mathbf{v}(\mathbf{r},t)}{\partial t} =& -\nabla\cdot\left[\mathbf{v}(\mathbf{r},t)\mathbf{v}(\mathbf{r},t)H(\mathbf{r},t)\right] \\ & -gH(\mathbf{r},t)\nabla h(\mathbf{r},t)\end{aligned} \tag{2.2}$$

where g is the strength of gravity and $h(\mathbf{r},t)$ is the height deviation from sea level. For discrete time steps of dt, the equations become

$$H(\mathbf{r},t+dt) = H(\mathbf{r},t) - \nabla\cdot\left[\mathbf{v}(\mathbf{r},t)H(\mathbf{r},t)\right]dt \tag{2.3}$$

$$\begin{aligned}H(\mathbf{r},t+dt)\mathbf{v}(\mathbf{r},t+dt) =& H(\mathbf{r},t)\mathbf{v}(\mathbf{r},t) \\ & -\nabla\cdot\left[\mathbf{v}(\mathbf{r},t)\mathbf{v}(\mathbf{r},t)H(\mathbf{r},t)\right]dt \\ & -gH(\mathbf{r},t)\nabla h(\mathbf{r},t)dt\end{aligned} \tag{2.4}$$

The Regional Ocean Modeling System (ROMS) simulates tsunamis by solving these equations using a finite difference method. Since this simulation is verified and based on different underlying mechanics, we have chosen it for comparison tests.

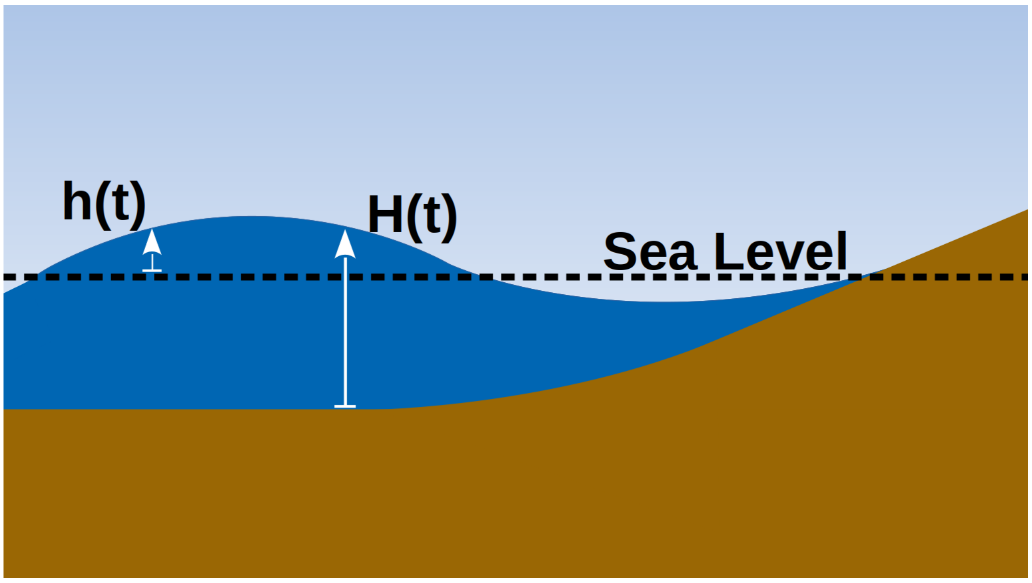


Figure 2.1: How water heights $H(\mathbf{r},t)$ and $h(\mathbf{r},t)$ are defined.

2.3 Tsunami Squares Model

Tsunami Squares equivalently solves these equations by using a new approach. First an area of water is split up into a grid of squares of length d, each with a height $H_i(t)$, velocity $\mathbf{v}_i(t)$, and acceleration $\mathbf{a}_i(t)$. Instead of solving the differential equations explicitly, each square of water is propagated individually over a time step dt according to the standard kinematic equations. The updated position of the water column being

$$\tilde{\mathbf{r}}_i(t) = \mathbf{r}_i + \mathbf{v}_i(t)dt + 0.5\mathbf{a}_i(t)dt^2 \tag{2.5}$$

and the updated velocity being

$$\tilde{\mathbf{v}}_i(t) = \mathbf{v}_i(t) + \mathbf{a}_i(t)dt \tag{2.6}$$

At each time step, the acceleration of each water column is calculated by summing the two types of acceleration: gravitational and frictional

$$\mathbf{a}_i(t) = \mathbf{a}_{grav,i}(t) + \mathbf{a}_{fric,i}(t) \tag{2.7}$$

The gravitational component is caused by any unevenness on the surface of the water. It is therefore expressed by a gradient term

$$\mathbf{a}_{grav,i}(t) = -\kappa g \nabla h_i(t) \tag{2.8}$$

where κ is a tuning parameter. The frictional acceleration term is as follows

$$\mathbf{a}_{fric,i}(t) = -\mu_d \frac{|\mathbf{v}_i(t)|}{H_i(t)} \mathbf{v}_i(t) \tag{2.9}$$

After the new position and velocity of each square of water is calculated, the volume and momentum of these squares are distributed proportionally to the squares it overlaps. For example, in Figure 2.2, each square is given its respective height, velocity, and acceleration at the start of the time step. For the case of square $i = 7$, the water's updated position is calculated and then the area overlap values δA_{7j} are calculated based on where the square lands. The momentum and volume of square seven is then split up among the four overlapping squares in proportion to the area overlap.

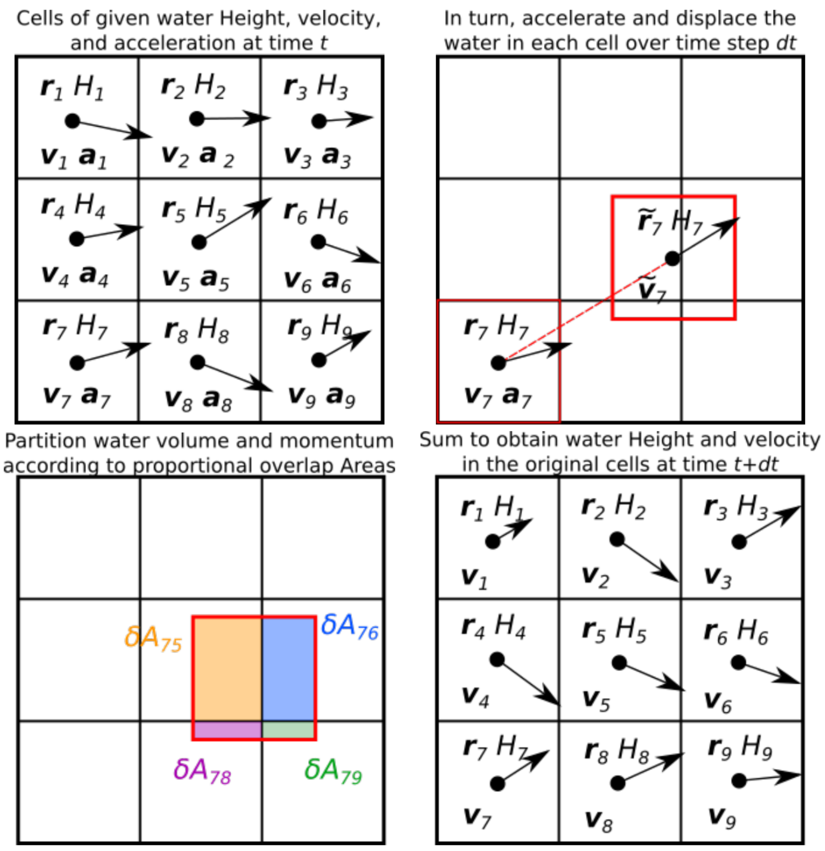

Cells of given water Height, velocity, and acceleration at time t
r1 H1 v1 a1
r2 H2 v2 a2
r3 H3 v3 a3
r4 H4 v4 a4
r5 H5 v5 a5
r6 H6 v6 a6
r7 H7 v7 a7
r8 H8 v8 a8
r9 H9 v9 a9
In turn, accelerate and displace the water in each cell over time step dt
r̃7 H7
ṽ7
r7 H7 v7 a7
Partition water volume and momentum according to proportional overlap Areas
δA75
δA76
δA78
δA79
Sum to obtain water Height and velocity in the original cells at time t+dt
r1 H1 v1
r2 H2 v2
r3 H3 v3
r4 H4 v4
r5 H5 v5
r6 H6 v6
r7 H7 v7
r8 H8 v8
r9 H9 v9

Figure 2.2: The Tsunami Squares algorithm. Shown is one time step for one square. (Wilson et al., 2020)

After calculating the new position and velocity of a given square, the amount of mass given to each of the overlapping squares is

$$H_j(t + dt) = \sum_i H_i(t)\delta A_{ij} \tag{2.10}$$

Similarly, the momentum is distributed as such

$$\begin{aligned}\mathbf{p}_j(t+dt) &= \sum_i \mathbf{p}_i(t)\delta A_{ij} \\ &= \sum_i \rho_w g A_i H_i(t)\mathbf{v}_i(t)\delta A_{ij}\end{aligned} \tag{2.11}$$

The velocity of each square at the next time step is

$$\mathbf{v}_j(t+dt) = \frac{\mathbf{p}_j(t)}{\rho_w g A_i H_j(t)} \tag{2.12}$$

These equations enforce volume and momentum conservation since the heights and momentum of each square are simply being redistributed.

$$\sum_i^N H_i(t+dt) = \sum_i^N H_i(t) \tag{2.13}$$

$$\sum_i^N \mathbf{p}_i(t+dt) = \sum_i^N \mathbf{p}_i(t) \tag{2.14}$$

A key feature of this water propagation technique is that it does not distinguish between wet and dry squares. This means water can be propagated from sea to land trivially. If water is transferred onto a dry square with $H = 0$, this square is simply given a non zero H, $\mathbf{a}$, and $\mathbf{v}$. This is particularly useful for mapping inundated regions because all one would have to do is track which dry squares became wet throughout the simulation. Other tsunami simulators do not have this capability and would have to use a separate model to estimate inundation.

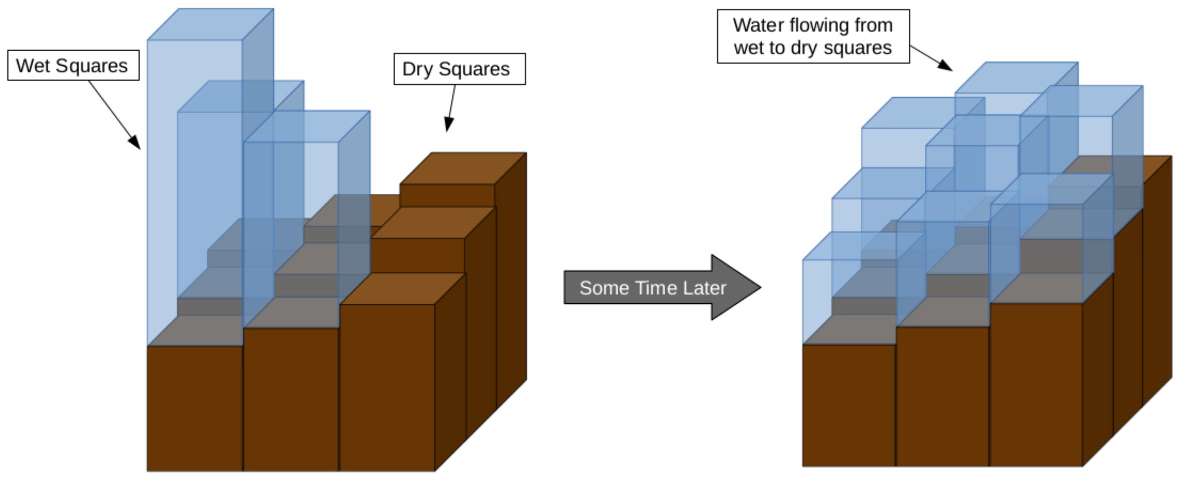


Figure 2.3: A hypothetical scenario showing water propagating from wet squares to dry squares. Tsunami Squares does not give special treatment to dry squares, making this trivial.

2.4 Smoothing Algorithm

No matter how well the algorithm works on its own, sharp changes in bathymetry, effects near land, and non-smooth waveforms will cause errors that will grow exponentially over time. Therefore, after the Tsunami Squares algorithm has been carried out for each square, a smoothing algorithm is then applied to even out any anomalies that will give rise from these nonphysical effects.

As for how the smoothing algorithm is carried out, it in essence takes some water from higher squares and transfers it to the neighboring lower squares. Figure 2.4 depicts a simplified 1D example containing only two squares for the purpose of demonstration.

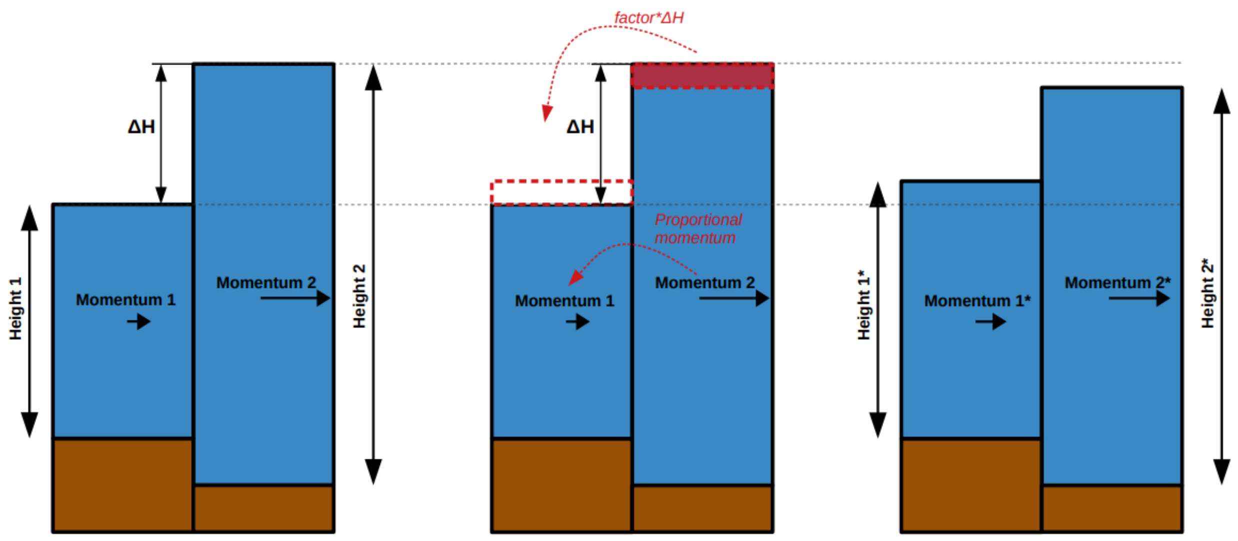


Figure 2.4: The smoothing algorithm showing the case for two columns of water. This is done by taking water from the right column (the higher column) and moving it to the left column (the lower column). Left: the columns before smoothing is applied. Middle: the algorithm in progress. Right: the final result.

The smoothing is done for both the height and velocity of each square. The amount of water transferred is proportional to the difference in height of the two squares. The factor f determines how much of that height difference is transferred from one square to the other. In Figure 2.4, this is depicted by the shaded red part in the center picture. A quantitative discussion of the optimal smoothing factor f is discussed in a further section. The heights and velocities transferred are

$$h_{\text{transfered}} = (h_2 - h_1) \cdot f \tag{2.15}$$

$$v_{\text{transfered}} = (v_2 - v_1) \cdot f \tag{2.16}$$

where h_1 and h_2 and the heights relative to sea level and the factor f is the factor chosen by the user. The higher the value of f, the greater the smoothing.

The following result shows the overall smoothing effect of the smoothing algorithm described above.

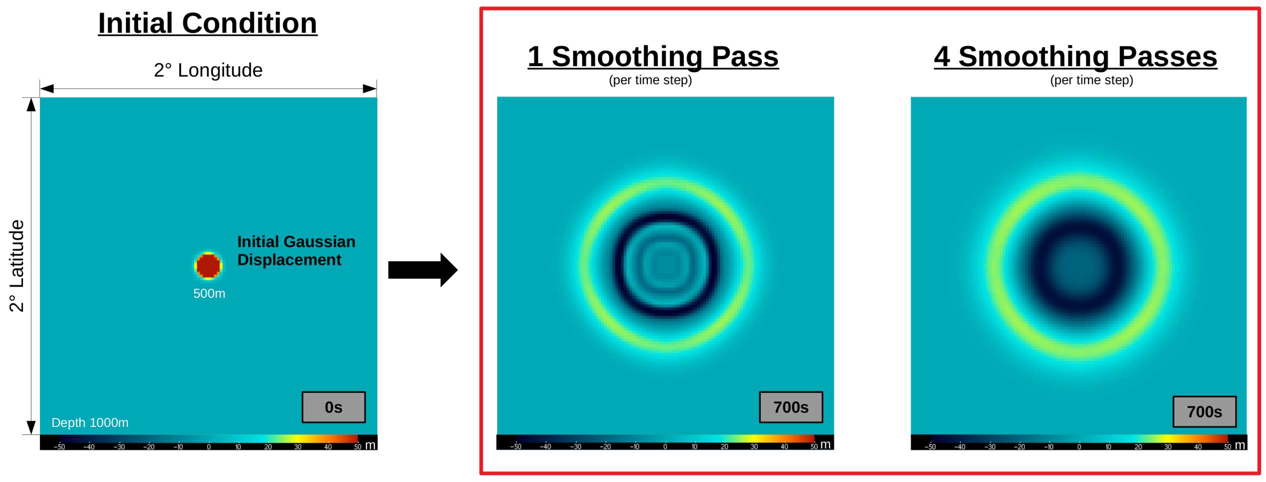


Figure 2.5: Circular wave showing the effects of increasing the number of smoothing passes. Left most image shows the 500 m tall Gaussian initial displacement. Center image shows the Gaussian displacement propagated over 700 seconds, smoothing once per time step. Right most image shows the result of smoothing four times per time step.

Figure 2.5 demonstrates the effect of one and four smoothing passes on a circular wave. More smoothing passes generally produce a more stable and simple result at the cost of losing some of the features of the waveform. These two effects then need to be balanced by adjusting the smoothing parameter f.

2.5 Code Structure

Originally written in Fortan, the code was rewritten in C++ for its ability to be parallelized, enhancing the speed of the simulation (Wilson et al., 2020). The input parameter files, code structure, and output file are shown in the following figure.

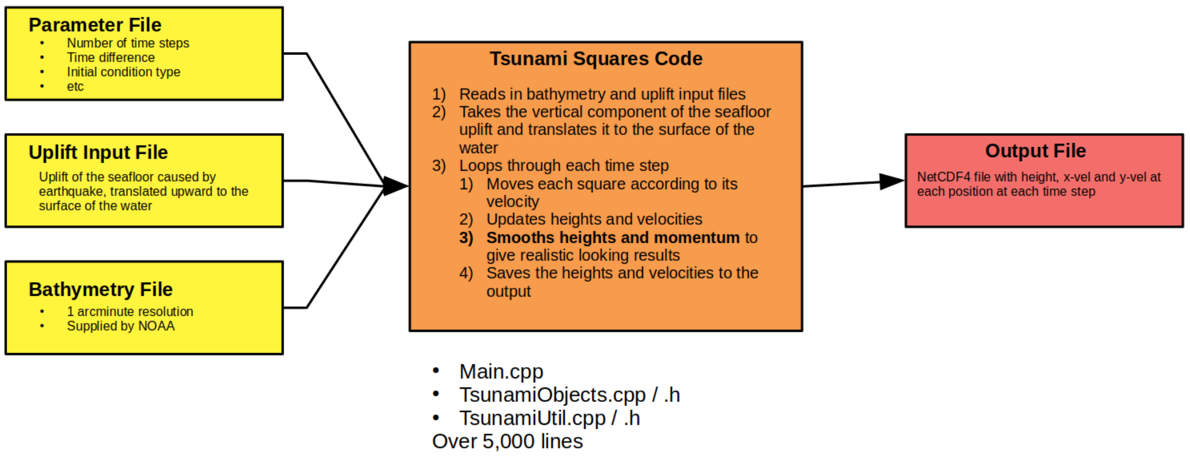

Parameter File
Number of time steps
Time difference
Initial condition type
etc
Uplift Input File
Uplift of the seafloor caused by earthquake, translated upward to the surface of the water
Bathymetry File
1 arcminute resolution
Supplied by NOAA
Tsunami Squares Code
1) Reads in bathymetry and uplift input files
2) Takes the vertical component of the seafloor uplift and translates it to the surface of the water
3) Loops through each time step
1) Moves each square according to its velocity
2) Updates heights and velocities
3) Smooths heights and momentum to give realistic looking results
4) Saves the heights and velocities to the output
Output File
NetCDF4 file with height, x-vel and y-vel at each position at each time step
Main.cpp
TsunamiObjects.cpp / .h
TsunamiUtil.cpp / .h
Over 5,000 lines

Figure 2.6: Structure of the Tsunami Squares code. A parameter file, initial condition file, and bathymetry file are supplied to the tsunami squares code where it transports each square individually at each time step, producing an output file in a netCDF4 format.

The process of producing a tsunami simulation using Tsunami Squares starts by preparing three files: the parameter file, the uplift initial condition file, and the bathymetry file. The bathymetry file supplies the simulator with a topography map over which to simulate and can be found in 60 arc-second, three arc-second, or one arc-second resolutions supplied by NOAA. NOAA's high resolution bathymetry data is specific to the US coast but other high resolution data can be found from other countries. The uplift initial condition file refers to the uplift generated by a given earthquake and may be the most difficult to obtain. If supplied the slip map to a given earthquake, the values for each slip value of the fault can be input into a separate script supplied by the Tsunami Squares code, generating the uplift file. However any other third party software can be used in the generation of the uplift file. The parameter file is a standard JSON file which supplies the simulation with information such as the number of time steps, time step, number of smoothing passes, file names, etc.

The body of the code consists mainly of three files totaling over 5,000 lines. These include the main file, a file which defines the objects and algorithms, and a file containing various utility functions. It starts by reading in the input files in order to set up the environment and define the initial condition. The program then loops over each time step, accelerating and moving each square individually, at which point overlaps are calculated. Once the new heights and velocities are determined through a sum of all contributions from overlaps for each square, both the heights and velocities are smoothed. The heights and velocities are then saved to that particular time step. After looping through all time steps, an output netCDF4 file is created to store the data.

2.6 Inundation Testing

To show the modified algorithm's raw inundation capabilities, runup on a plane beach was measured for four beach slopes of differing angles. However, there should not be any significant difference between the old algorithm and the new algorithm since flows onto land remain largely the same. For these tests, no change in treatment was given to the water on land. The same rules that were applied to the open water were applied to the beach. This gives a starting point for future development and improvement.

Solitons were used as the initial condition as they have a permanent form and defined width for every amplitude (A) and depth (d). It is defined as follows:

$$h(x) = A \cdot \operatorname{sech}^2(kx), \quad k = \frac{1}{d}\sqrt{\frac{3A}{4d}} \tag{2.17}$$

Figure 2.7 shows an example of the setup for one such test.

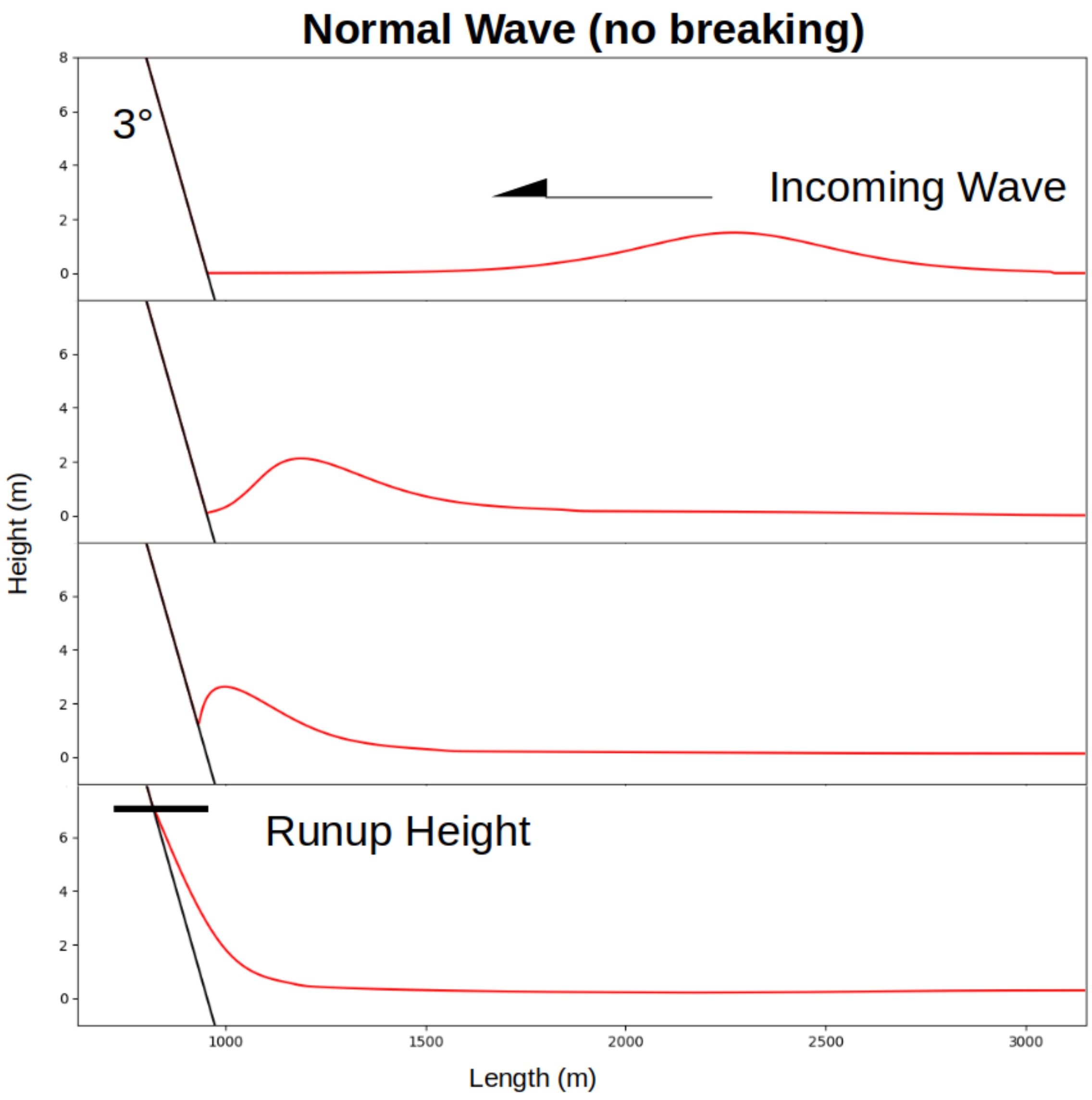


Figure 2.7: The setup of the maximum runup height tests. An incoming soliton wave travels to the left and approaches a gradual slope (Top). As the wave strikes land it travels up the slope a maximum distance (bottom). This maximum distance is recorded for the given angle and initial wave height.

The values for runup height, amplitude, and depth are not important in comparison to their ratios. The amplitude to depth ratio gives the relative size of the wave, and the runup height to amplitude ratio gives the relative runup height. These two ratios fully describe each scenario and should give equivalent results regardless of the absolute values of the parameters. Expected runup heights are given by the following equation (Synolakis, 1987).

$$R = 2.831A \cdot (\cot\theta)^{\frac{1}{2}} \left(\frac{A}{d}\right)^{\frac{1}{4}} \tag{2.18}$$

The breaking limit is given by this formula (Synolakis, 1987).

$$\frac{A}{d} < 0.48(\tan\theta)^{\frac{10}{9}} \tag{2.19}$$

where the waves will not break as long as this inequality holds. They are denoted as a vertical markers in the following figure. Equation 2.18 is expected to hold up until this breaking point.

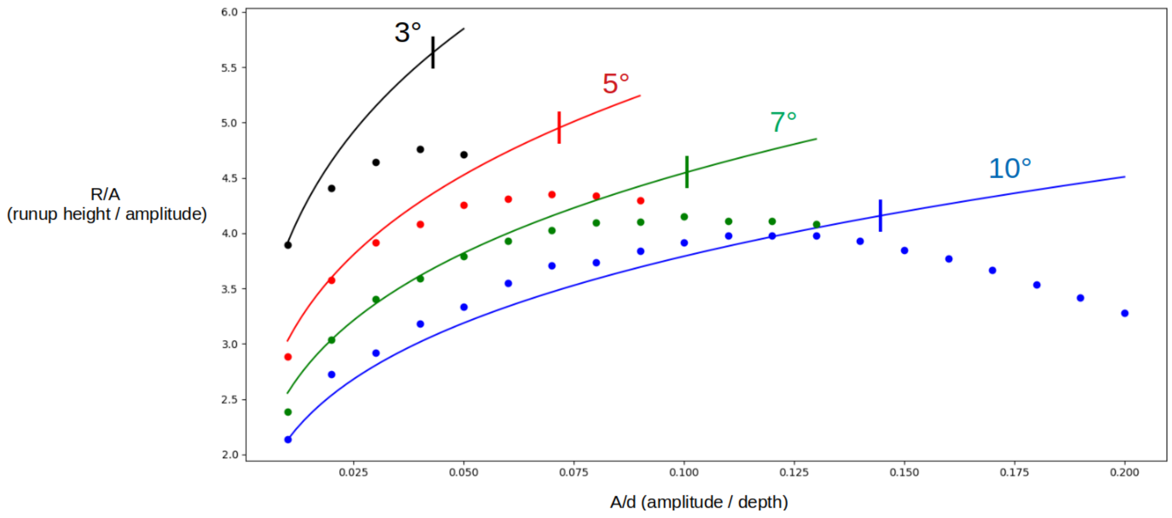


Figure 2.8: Results showing various beach slopes and incoming wave amplitudes where vertical lines indicate the wave breaking limit. Generally, results are more accurate for steeper slopes. Results after the breaking limit are particularly inaccurate.

Results from Figure 2.8 show good alignment for small amplitudes. Steeper angles show accurate results up until the breaking point. At this point and beyond, the wave begins to steepen, decreasing energy and resulting in a lower runup height. As for shallower angles, the wave experiences non-linear effects which deform the wave shape, steepening the leading edge. This is one known reason for the observed inaccuracy in Figure 2.8.

Since this was merely an initial test of the modified algorithm's inundation accuracy, results are acceptable. However, there is room for improvement for shallower beach slopes. Regarding scenarios using realistic bathymetry and initial conditions from real events, amplitude to depth ratios are usually extremely small ($\sim$0.005) and runup slopes can be very steep ($\sim$10 degrees) near most of the particularly mountainous Pacific coast. Using these realistic conditions, TS is in agreement with the analytical results shown in Figure 2.8.

Chapter 3

Modifications to The Tsunami Squares Algorithm

3.1 Introduction

Tsunami simulators are typically faced with a trade-off between accuracy and computational cost. An increase in the resolution or accuracy in a given tsunami simulation is usually compensated by an increase in computational cost. This poses several obvious problems for machine learning applications due to the computational resources needed in the creation of large pre-computed tsunami simulation databases. Recent machine learning applications include utilizing a convolutional neural network and a multilayer perceptron to estimate inundation in real time by linking low resolution simulations calculated in real time to pre-computed high resolution inundation maps (Fauzi & Mizutani, 2020). Other work has utilized a matching scheme using principal component analysis to link a similar database of low and high resolution simulations (Mulia et al., 2018, 2020). Other authors have used stochastic earthquake source models for the Nankai-Tonankai earthquake to quantify the uncertainty of a tsunami impact at coastal regions (Goda et al., 2018). However, the work mentioned uses a collection of pre-computed tsunami source models only on the order of hundreds. Adding fault scenarios is essential to improve the effectiveness of these techniques as problems arise if there is no acceptable match in the database (Mulia et al., 2018, 2020; Fauzi & Mizutani, 2020). However, when increasing the number of simulations in the database, scenarios must be unique to maintain a statistical correlation between low resolution and high resolution simulations (Mulia et al., 2018, 2020).

Presented in this chapter are improvements made to Tsunami Squares, a computationally lightweight tsunami simulator which utilizes a cellular automata technique, leaving it well suited for these tsunami database creation applications. The changes made consist of two small changes to how these individual squares of water are accelerated and transported. In addition, a new technique has been developed to ensure energy conservation. These changes allow Tsunami Squares to simulate waves with enhanced accuracy and resolution, allowing for use in database creation applications such as machine learning, where each tsunami in the database requires sufficient waveform resolution in order to distinguish between other tsunamis in the database.

3.2 Tsunami Squares Modifications

The modifications made to Tsunami Squares are mainly related to how acceleration and distance traveled are calculated. First we will discuss the specifics of how these two quantities are defined in the algorithm.

At the beginning of each time step, the acceleration of each square is calculated. It is caused by any unevenness on the surface of the water and is therefore expressed by a gradient term

$$\mathbf{a}_i(t) = -g\nabla h_i(t) \tag{3.1}$$

Once this is determined through a calculation of the slope at that point, the final position can be calculated. However, the slope and final position can be computed multiple ways. Many experiments suggest that new methods to calculate these two quantities yield better performance. This means greater accuracy, improved energy conservation, and more detail contained in the waveform. We will now discuss the differences in implementation and how this impacts the resulting wave.

Previous implementations of Tsunami Squares have calculated the slope by using the central difference approximation. For a 1D wave in the x-direction this is

$$\mathrm{S} = \frac{h_{i+1} - h_{i-1}}{2\Delta x} \tag{3.2}$$

where Δx is the width of the square and S is the slope. The final distance is then calculated according to standard kinematics

$$\tilde{\mathbf{r}}_i(t) = \mathbf{r}_i + \mathbf{v}_i(t)dt + \tfrac{1}{2}\mathbf{a}_i(t)dt^2 \tag{3.3}$$

To compare, the current implementation of Tsunami Squares calculates the slope in the following way. It first considers the direction of the individual square-v (square velocity), then takes the slope by considering only itself and the square in that direction. For example in 1D, if square_v points to the right, the slope is calculated by only considering the current height h_i and the height to right of it h_{i+1}.

$$\mathrm{S}_{\text{square\_}v>0} = \frac{h_{i+1} - h_i}{\Delta x} \tag{3.4}$$

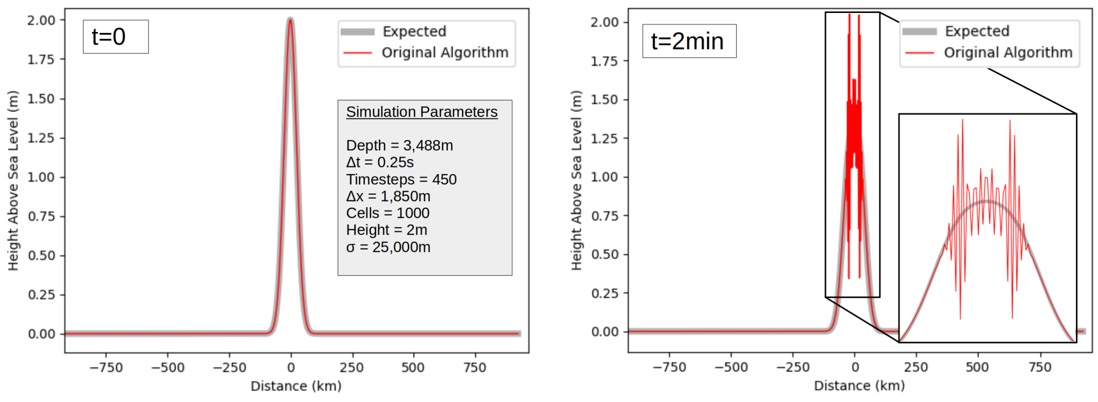
$$\mathrm{S}_{\text{square\_}v<0} = \frac{h_i - h_{i-1}}{\Delta x} \tag{3.5}$$

The final distance is then calculated by

$$\tilde{\mathbf{r}}_i(t) = \mathbf{r}_i + \mathbf{v}_i(t)dt + \mathbf{a}_i(t)dt^2 \tag{3.6}$$

which differs from the previous implementation by a factor of $\frac{1}{2}$ in the term containing acceleration. The new distance formula is equivalent to transporting the square according its final velocity $(\mathbf{v}_i(t) + \mathbf{a}_i(t)dt)$ instead of the average velocity as is used in standard kinematic formulas.

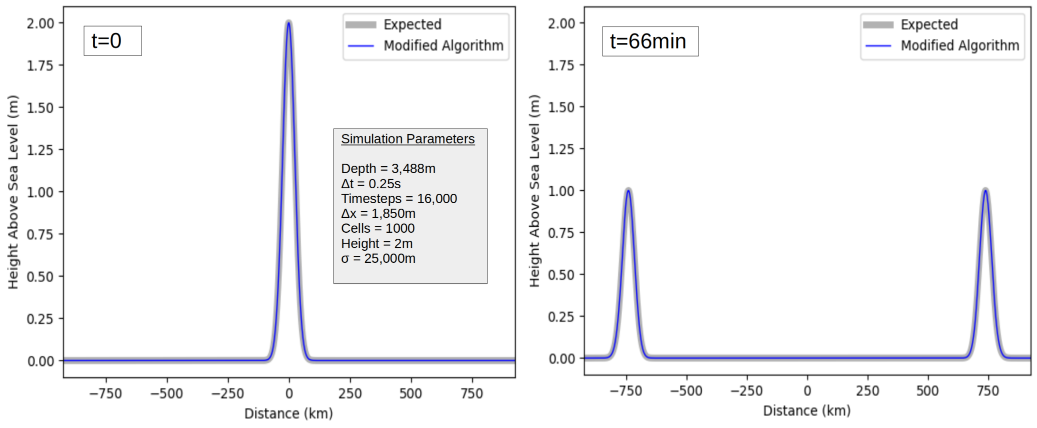
A 1D stationary Gaussian pile of water will be simulated and allowed to propagate left and right to show the difference between the original implementation and the new implementation. An expected solution is shown by solving the linear shallow water equations using a finite difference method. We present results using the original implementation defined by Equations 3.2 and 3.3 followed by the results using the new implementation defined by Equations 3.4, 3.5 and 3.6.



Simulation Parameters

Depth = 3,488m
Δt = 0.25s
Timesteps = 450
Δx = 1,850m
Cells = 1000
Height = 2m
σ = 25,000m

Figure 3.1: A 1D example of the original Tsunami Squares implementation using Equations 3.2 and 3.3 starting from an initial stationary Gaussian pile of water (left). The wave propagates for a few time steps before experiencing erroneous behavior (right). Even with smooth initial conditions and small Δt, the original Tsunami Squares implementation cannot stably propagate waves.



Simulation Parameters

Depth = 3,488m
Δt = 0.25s
Timesteps = 16,000
Δx = 1,850m
Cells = 1000
Height = 2m
σ = 25,000m

Figure 3.2: A 1D example of the modified Tsunami Squares implementation using Equations 3.4, 3.5, and 3.6 starting from an initial stationary Gaussian pile of water (left). The wave then propagates in alignment with the expected analytical result for 16,000 time steps (right).

Results show clearly that the new way of calculating the slope (Equations 3.4 and 3.5) in combination with the new way of calculating the final position (Equation 3.6) allows the wave to propagate in exact agreement with the expected solution. Of course, these results are under ideal conditions (small Δt and a smooth waveform). If one is to move on to a simulation with conditions that are not ideal, such as in virtually every realistic tsunami scenario, one needs a way to smooth out the errors that will eventually arise due to sudden changes in bathymetry, effects at shore, and non-ideal waveforms. In the original implementation, such a smoothing algorithm is necessary for the wave to function properly for any condition.

3.3 Smoothing

No matter how well the algorithm works on its own, sharp changes in bathymetry, effects near land, and non-smooth waveforms will cause errors that will grow exponentially over time. Therefore, after the Tsunami Squares algorithm has been carried out for each square, a smoothing algorithm is then applied to even out any anomalies that will give rise to these nonphysical effects. To demonstrate the overall effects the smoothing algorithm has on a wave that is experiencing erroneous behavior, smoothing has been applied to the wave produced by the original algorithm in Figure 3.1.

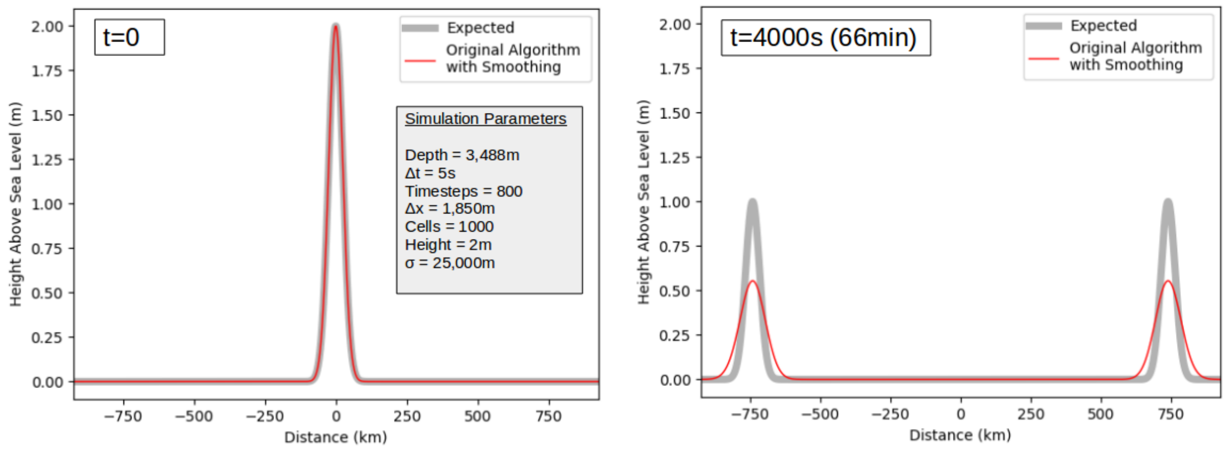


Figure 3.3: The 1D Tsunami Squares simulation from figure 3.1 with smoothing (left). The waves then propagate left and right after which the clear effects of smoothing can be seen (right). It results in a wave with a lower amplitude, greater width, and lower energy (55% of initial).

The wave that is produced by the original simulation has errors, but these are removed or reduced by the smoothing procedure, allowing the waves to propagate virtually error-free to left and right. However, as can be seen, the smoothing algorithm gives rise to several unwanted effects. These are listed below.

- Reduced amplitude
- Increased width

- Reduced energy

It is therefore the case that *less smoothing is better*. This then answers the question of why it is more beneficial to use the new implementation over the original implementation, when in practice both require smoothing. It is because the new implementation needs significantly less smoothing, and will therefore produce fewer of these undesired effects. The original implementation needs smoothing to operate, whereas the new implementation merely needs it for maintenance as it can function to a good degree on its own.

Even with the little amount of smoothing needed by the new implementation for maintenance, over large distances these undesirable effects come into play. Most notably, energy is lost. To conserve energy, it must be manually added back in. Multiple techniques can be used to accomplish this and will be discussed in the next section.

3.4 Adding Energy

After changes were made to the Tsunami Squares algorithm detailed in Equations 3.4, 3.5, and 3.6, waves can now propagate on their own with enhanced stability (see figure 3.2). However, although the modified algorithm is now stable under ideal conditions (smooth waveform, small time steps, flat seafloor), non-ideal conditions cause the wave to experience errors that grow over time. To remedy this, a smoothing algorithm was introduced which acts as a moving average applied across all squares in the simulation. This has unfortunate side effects of decreasing the amplitude and energy over time. The solution to these undesired effects is to manually add energy back into the simulation. A new technique used to accomplish this will be described in this section.

The technique described here has the effect of “sharpening” the wave. This means the amplitude of the waveform increases over time and width of the wave decreases over time.Figure 3.4 shows the desired effect.

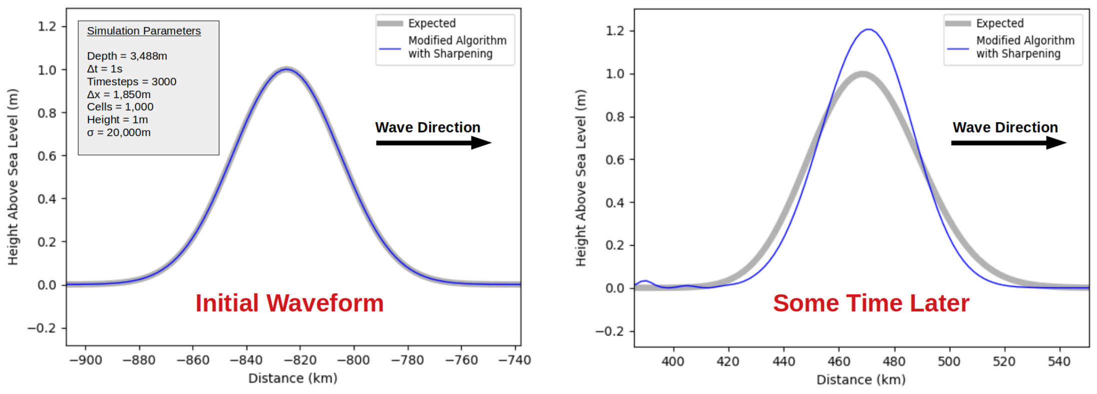


Figure 3.4: Blue: A positive wave showing the effect of sharpening. Gray: Finite difference solution. Due to the sharpening effect, the wave's width decreases and its height increases over time, increasing the energy substantially as well.

The sharpening effect shown in Figure 3.4 contrasts the smoothing effect shown in Figure 3.3. If balanced correctly, the combination of these two effects leaves the original waveform unchanged, while at the same time smoothing out any anomalies. The parameter tuning necessary to produce such a balance will be discussed later in this section.

The sharpening effect is accomplished by shifting the acceleration towards the opposite direction of wave-v (wave velocity) by a small amount. This amount is denoted by a, which is defined by

$$a = \frac{(\text{distance moved})}{\Delta x} \tag{3.7}$$

where Δx is the square length. $a = 1.0$ would therefore shift the acceleration by one full square length or one Δx. Figure 3.5 shows this process.

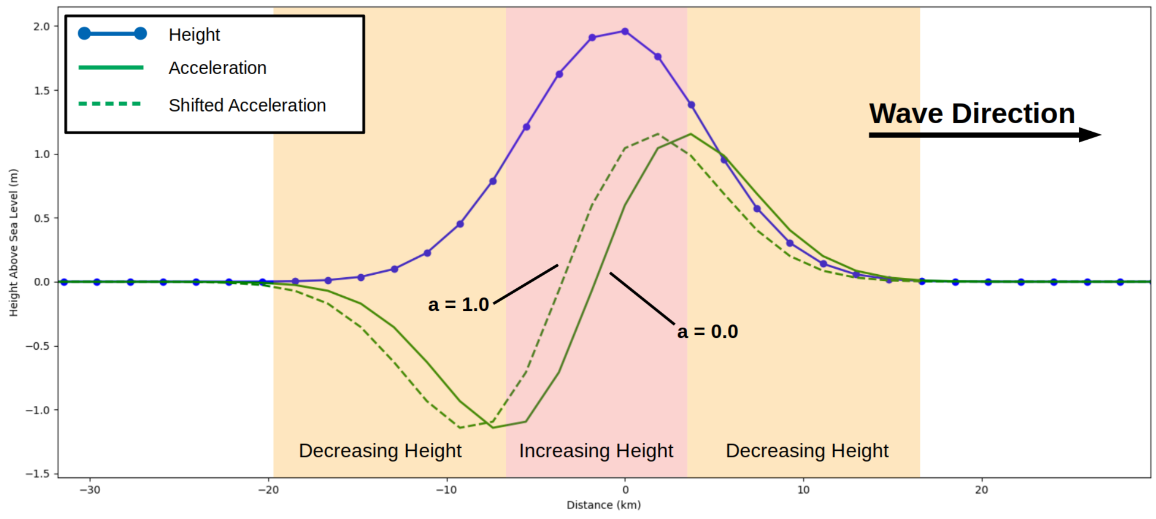


Figure 3.5: Diagram showing how a shift in the acceleration curve will cause an increase in the acceleration in the center (red shaded region) and a decrease in acceleration in the outer regions (orange shaded regions). An increase in acceleration leads to an increase of velocity which in turn increases the height. This acceleration shift ultimately leads to a sharpening effect on the wave, increasing its energy.

By shifting the acceleration curve, acceleration is either raised or lowered from its original value. The more the acceleration is shifted, the greater this increase or decrease. Regions where the acceleration is raised cause a rise in height and regions where the acceleration is lowered causes a decrease in height. This is indicated by the red and orange shaded regions in Figure 3.5. An increase in height follows from an increase in acceleration because as the acceleration rises, so do the square velocities, causing the square heights to increase as well.

Figure 3.6 below will be used to describe exactly how the slope is taken with this newly added wave-v directional dependency.

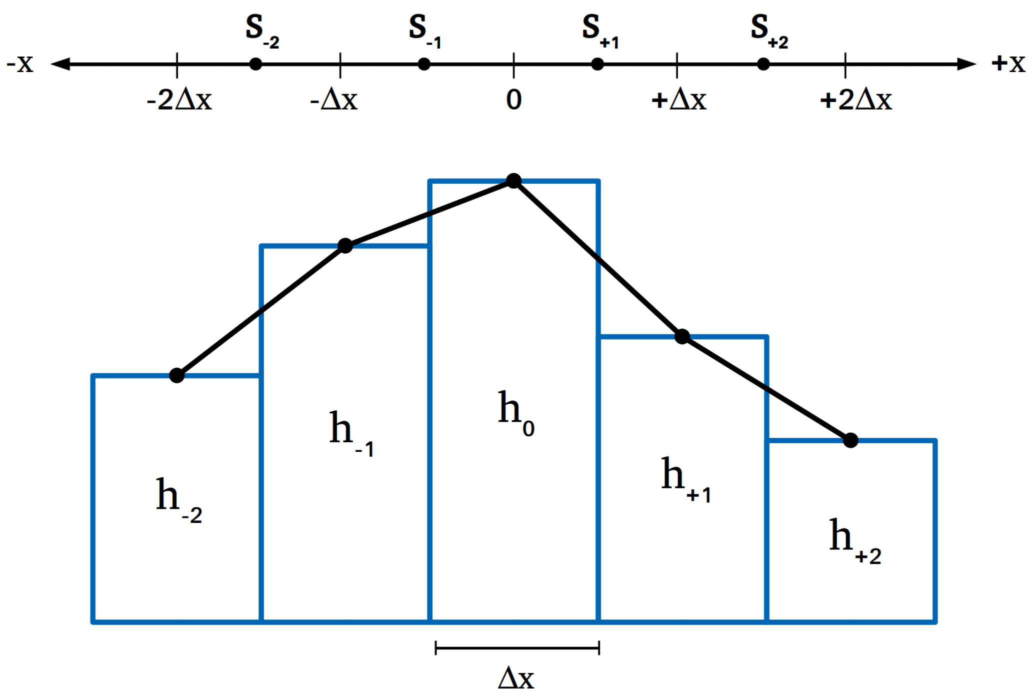


Figure 3.6: A 1D example showing positions at which the slope is well defined (S_{-2}, S_{-1}, S_{+1}, S_{+2}). Slopes at all positions in between these points can calculated by linear interpolation.

Figure 3.6 shows distance from the current square and the slopes which are defined at certain locations (S_{-2}, S_{-1}, S_{+1}, S_{+2}). We can further define the slope at every position by a simple linear interpolation. This gives us the opportunity of shifting the acceleration (or equivalently shifting the slope) at intervals smaller than Δx.

Since the acceleration is to be shifted based on the direction of wave-v, one piece of information must first be made clear. In general for well behaved waves, wave-v points in the same direction as square-v if h is positive and wave-v points in the opposite direction as square-v if h is negative. So by looking at square-v and the height, which is well defined for every square, the wave-v direction can be estimated to good approximation.

To demonstrate the process of determining the slope, an example will be given. Say we want to shift the acceleration by $0.25\Delta x$ (a common and well performing choice) for the wave given in Figure 3.5. The first step is to consider square-v to determine if S_{-1} or S_{+1} will be used. This slope is the fundamental algorithm's slope detailed in Equations 3.4 and 3.5 and acts as the starting point before any shifting has occurred. After checking that the sign of square-v is positive, S_{+1} is taken.

To shift the acceleration, the wave-v direction must be known. It can be estimated that for a square with positive h and positive square-v, wave-v must also be positive. Since we need to shift the slope towards the opposite direction of wave-v, the slope must be taken that distance towards the direction of wave-v. We now know we want to take the slope $0.25\Delta x$ to the right of S_{+1}. We can therefore interpolate between S_{+1} and S_{+2} to find the desired shifted slope.

$$\begin{aligned} S_{\text{shifted}} &= S_{+1} + 0.25\Delta x \frac{(S_{+2} - S_{+1})}{\Delta x} \\ &= S_{+1} + 0.25 S_{+2} - 0.25 S_{+1} \\ &= 0.75 S_{+1} + 0.25 S_{+2} \\ &= (1-a)S_{+1} + (a)S_{+2} \end{aligned} \tag{3.8}$$

More generally, a is the fraction of Δx the wave is to be shifted. The shifted slope can be written for all combinations of h and square-v:

$$S_{\text{shifted}} = (1-a)S_{+1} + (a)S_{+2} \qquad (h > 0,\ \text{square-}v > 0) \tag{3.9}$$

$$S_{\text{shifted}} = (1-a)S_{-1} + (a)S_{-2} \qquad (h > 0,\ \text{square-}v < 0) \tag{3.10}$$

$$S_{\text{shifted}} = (1-a)S_{+1} + (a)S_{-1} \qquad (h < 0,\ \text{square-}v > 0) \tag{3.11}$$

$$S_{\text{shifted}} = (1-a)S_{-1} + (a)S_{+1} \qquad (h < 0,\ \text{square-}v < 0) \tag{3.12}$$

For a given simulation, choosing values within the range of $0.1 < a < 0.3$ has been tested to perform most accurately. For each value of a, a specific smoothing factor f (Equations 2.15 and 2.16) can be calculated to exactly cancel out the sharpening effects. This factor f was determined empirically and depends on a, Δx, Δt and height of water column H:

$$f = 0.24 \frac{\sqrt{gH} \Delta t \cdot a}{\Delta x} \tag{3.13}$$

By using a value of $0.1 < a < 0.3$ and the smoothing factor given above, the wave will now conserve energy and preserve shape while eliminating any erroneous behavior.

3.5 Additional Refinements to Improve Accuracy

The methods described thus far operate as they should for waves which contain only positive or only negative heights. For more complicated waveforms that include both positive components and negative components, problems arise. With the great reduction of smoothing due to the introduction of the new modifications and methods, errors that were previously smoothed over have now surfaced. These errors are shown in the following two figures.

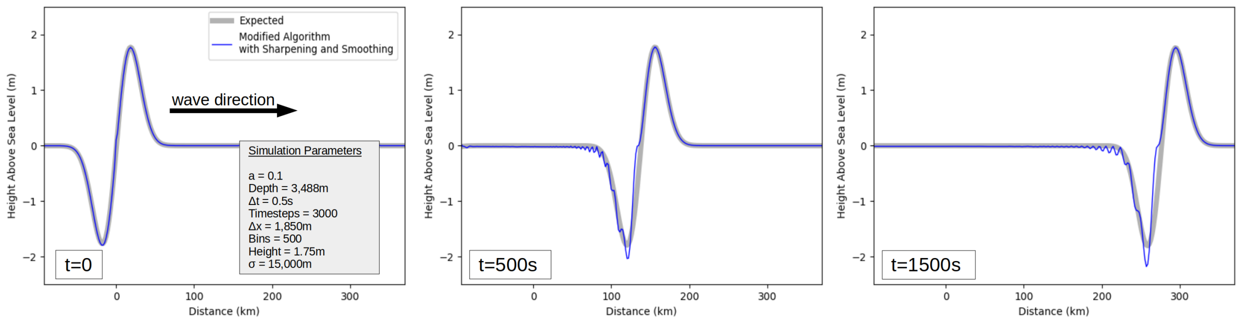


Figure 3.7: Errors at the boundary between a negative and positive wave, showing the case for a positive slope traveling to the right. This contains the case where there are two squares with square-v pointing away from each other.

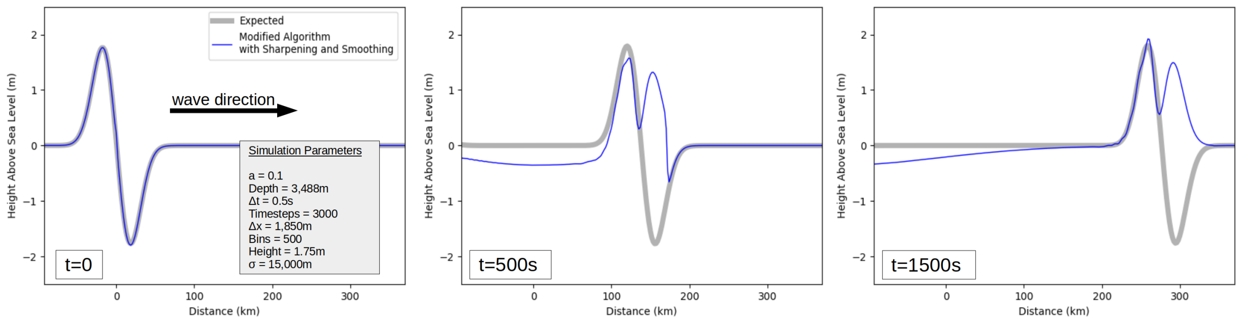


Figure 3.8: Errors at the boundary between a negative and positive wave, showing the case for a negative slope traveling to the right. This contains the case where there are two squares with square-v pointing towards each other.

The two types of errors shown are due to a change in the direction of square-v from one square to another. The error in the first image is caused by two squares at $h = 0$ which have square velocities pointing away from each other. The error in the second image is caused by two squares which have square velocities pointing inwards towards each other. In well behaved waves, water is exchanged uniformly from left to right or visa versa (indicated by the direction of square-v). If this direction changes there is a discontinuity in this exchange of water, leading to erroneous behavior as shown in the previous two images.

The solution to this problem involves modifying the acceleration and velocity of the two squares which have opposing square-v directions. Not only do these modifications depend on the square-v direction, but also the slope of h at that point (whether it is positive or negative). Therefore there are four total scenarios which must be treated with their own modifications. The four scenarios can be categorized as the cases of Figures 3.7 and 3.8 plus the cases where the two waves are traveling in the opposite direction. These rules are shown below.

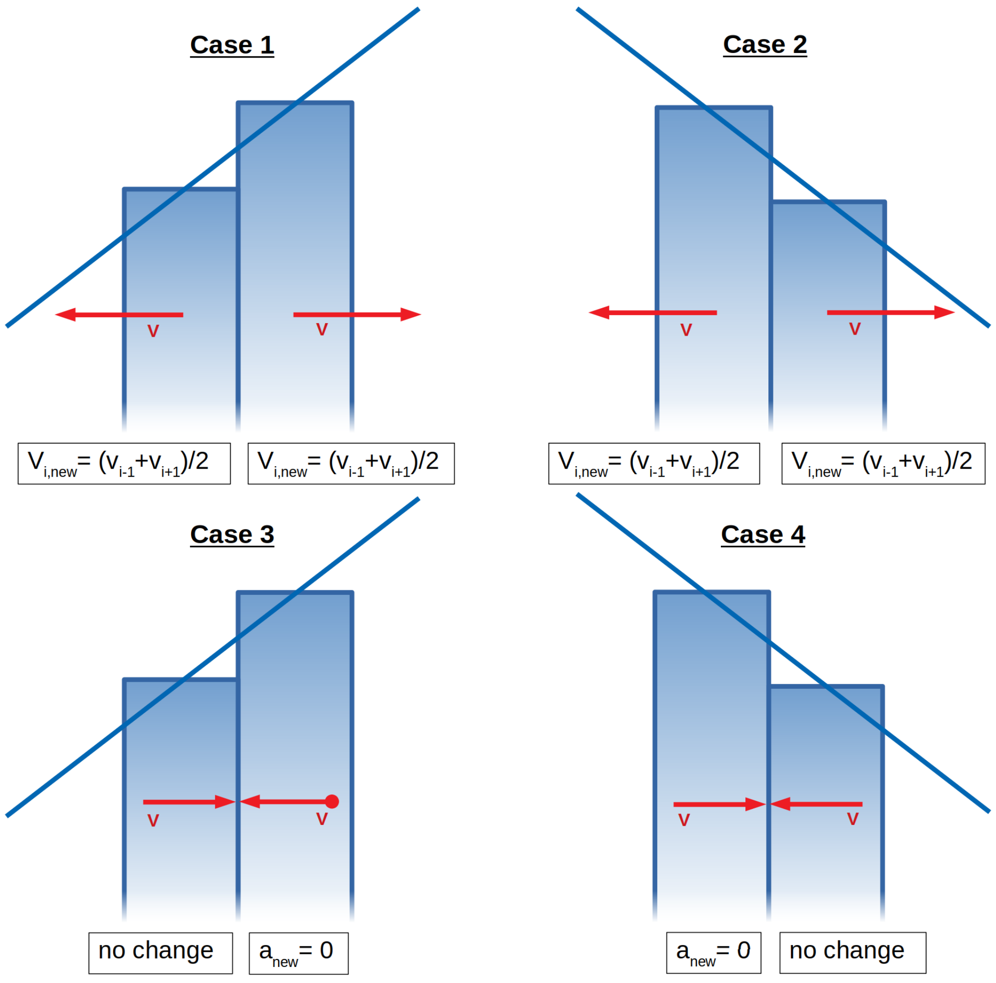

Case 1
v
v
V_{i,new}= (v_{i-1}+v_{i+1})/2
V_{i,new}= (v_{i-1}+v_{i+1})/2
Case 2
v
v
V_{i,new}= (v_{i-1}+v_{i+1})/2
V_{i,new}= (v_{i-1}+v_{i+1})/2
Case 3
v
v
no change
a_{new}= 0
Case 4
v
v
a_{new}= 0
no change

Figure 3.9: The changes to acceleration and velocity that must be applied to remedy the problems experienced at the boundary between positive and negative square-v shown in Figures 3.7 and 3.8. There are four scenarios relating to the four unique combinations of positive and negative height and square-v slopes present at the square-v = 0 boundary. Only the changed parameters are specified. If either acceleration or velocity is not mentioned, it is left unchanged.

The rules for each pair of squares are also given in the following table. For consistency, case one corresponds to the upper left portion of the table, case two corresponds to the upper right, and so on.

		Height Slope			
		Positive		Negative	
Velocity Slope	Positive	Left $v_{i,\text{new}} = \frac{(v_{i-1}+v_{i+1})}{2}$	Right $v_{i,\text{new}} = \frac{(v_{i-1}+v_{i+1})}{2}$	Left $v_{i,\text{new}} = \frac{(v_{i-1}+v_{i+1})}{2}$	Right $v_{i,\text{new}} = \frac{(v_{i-1}+v_{i+1})}{2}$
	Negative	Left no change	Right $a_{\text{new}} = 0$	Left $a_{\text{new}} = 0$	Right no change

Table 3.1: Rules to remedy problems located where a pair of squares crosses square-v = 0. Only the changed parameters are specified. If either acceleration or velocity is not mentioned, it is left unchanged. These rules are applied separately for both the x and y direction, where left and right correspond to bottom and top squares.

It can be seen that the rules involve setting the acceleration to zero for some squares and setting the velocity to the average velocity for others. With these rules implemented, the simulation can now run stably and reliably, free of the errors observed in Figures 3.7 and 3.8.

3.6 Comparison to a Previous Method of Energy Conservation

The previous method of adding energy used in previous versions of Tsunami Squares relied on a negative friction coefficient for the seafloor bottom, which is somewhat unrealistic. Total energy is monitored at every time step and if it falls below a threshold, a negative value of friction is applied as a way to input extra energy. Once an energy cap is reached, the friction coefficient is set to zero and the energy is allowed to gradually decrease.

Now we compare the negative friction method to the method devised here. For this test, we increase the complexity of the scenario by sending two waveforms towards each other and observe them after they have passed through each other. We use an analytical solution for a benchmark.

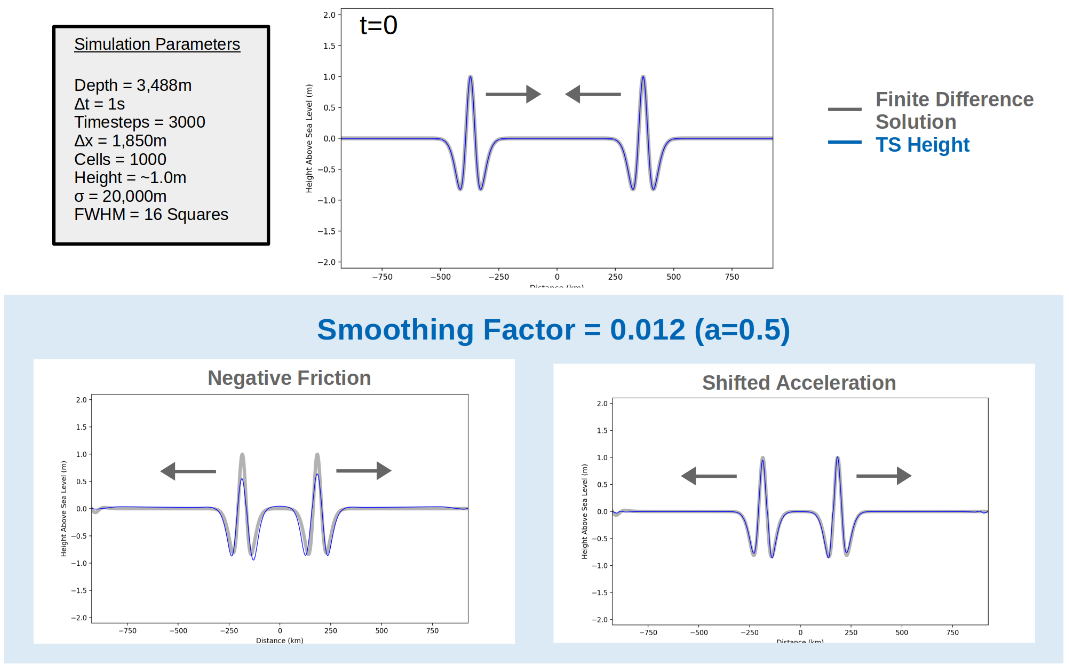


Figure 3.10: Comparison of two methods of conserving energy: the previous method using negative friction and the newly implemented shifted acceleration method. The top image shows the initial condition of two waveforms with wave-velocities aimed towards each other. The left image shows the negative friction results after these two waveforms have passed through each other. The right image shows the shifted acceleration method.

We can see from Figure 3.10 that the negative friction waveform has lower amplitudes than the expected, while the shifted acceleration waveform closely resembles the expected. Overall, the shifted acceleration method appears to outperform the negative friction method for complicated scenarios such as what one would expect to find in any full scale tsunami simulation.

3.7 Full Simulation Results

The following result compares a simulation using the original implementation compared to several simulations using the new implementation (each with different levels of smoothing). The event used in this comparison is the 2011 Tohoku event with an initial condition obtained by inverting wave gauge data (Song et al., 2012; Song, 2007).

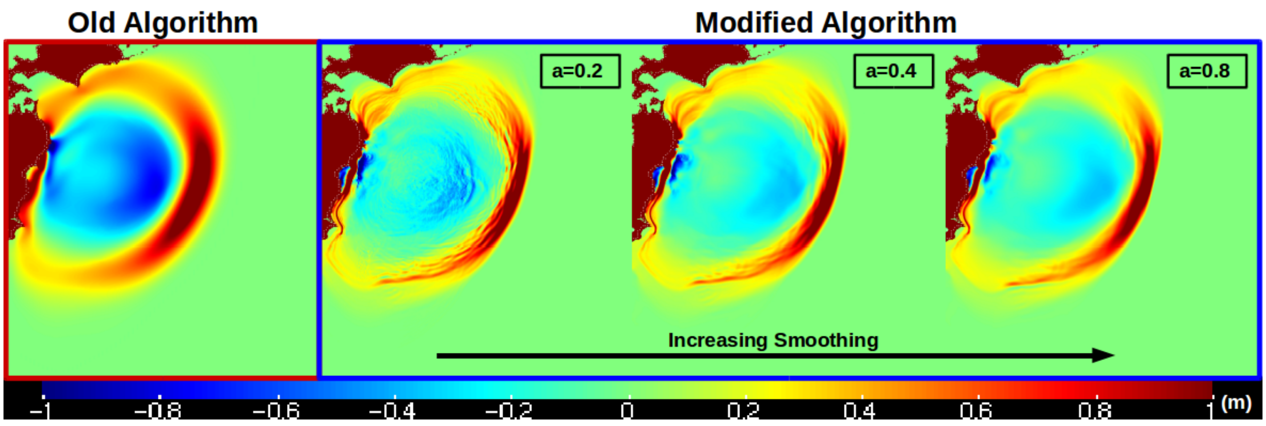


Figure 3.11: Far left: Results of the 2011 Tohoku event using the original implementation. Right three: Results using the new implementation, showing increasing amounts of sharpening (indicated by the value "a") corresponding to increasing amounts of smoothing.

The value a used in each simulation in Figure 3.11 is proportional to how much smoothing is used (Equation 3.13). It follows from this equation that as the value a increases, in turn so does the amount of smoothing used.

Compared to the original algorithm, the modified algorithm produces a simulation with enhanced detail. This is possible because of the significant reduction in the amount of smoothing necessary. To quantify, the smoothing factor (Equations 2.15 and 2.16) used in the new implementation is 5-15 times less than the original implementation.

For a more detailed comparison, buoy data from the 2011 Tohoku event and the 2010 Maule event were used to compare the original implementation, new implementation, and the Regional Ocean Modeling System (ROMS). ROMS is a verified wave simulator which solves three dimensional Reynolds-averaged Navier-Stokes equations using a finite difference method and has been applied to several tsunami studies (Haidvogel et al., 2008; Song et al., 2017). It is used in the comparison to give Tsunami Squares a reference to an accurate tsunami simulator which uses the similar initial conditions derived by inverting wave gauge data (Song et al., 2012; Song, 2007). It should be noted that the TS source time function is much simpler than the actual earthquake, which might be responsible for some deviation in the results. ROMS, however, uses a more sophisticated source time function.

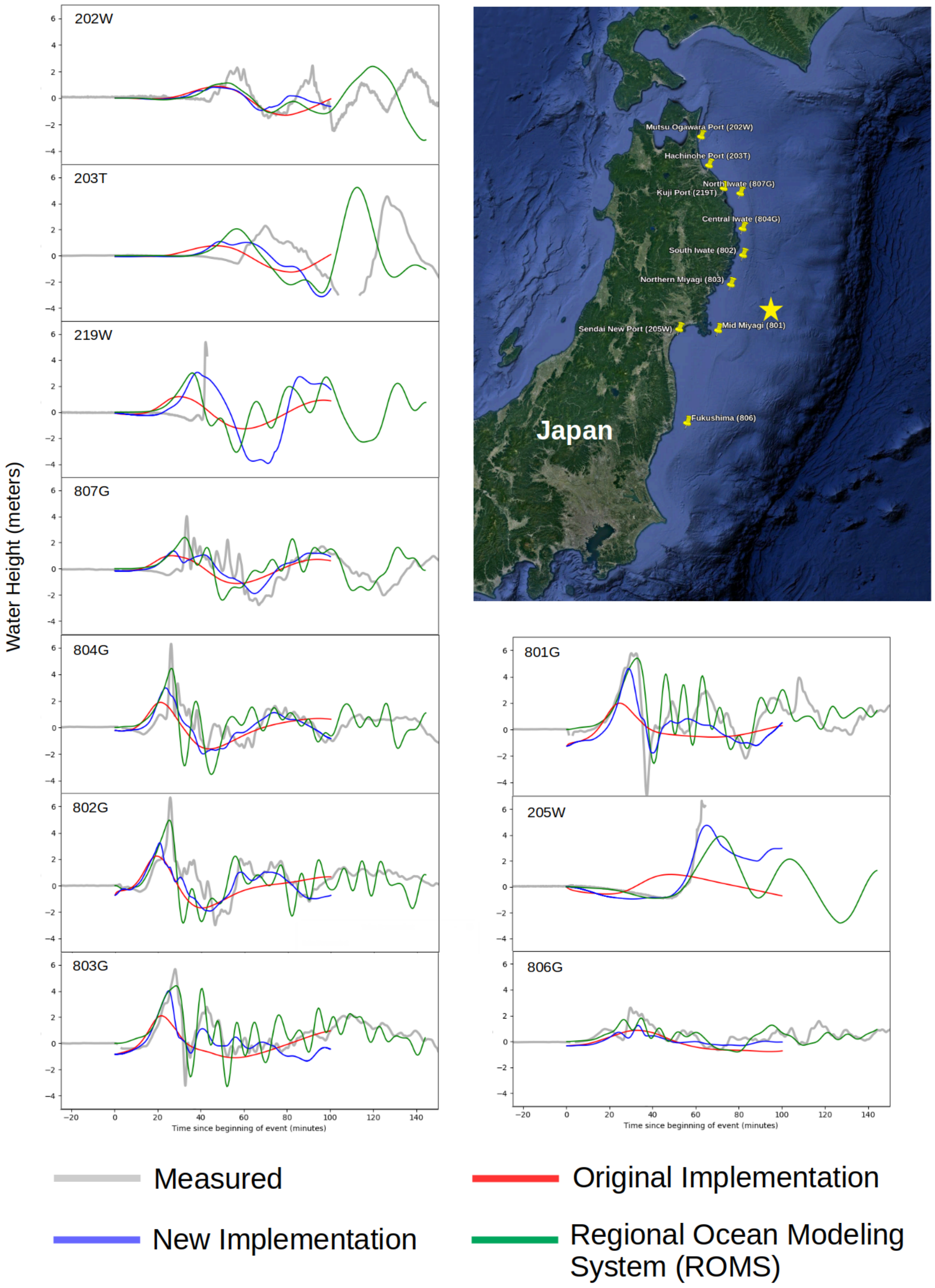


Figure 3.12: Buoy comparison plots for the 2011 Tohoku Tsunami. The simulation using the new implementation uses an a value of 0.4. Results show better alignment and more detail for the new implementation in nearly all cases.

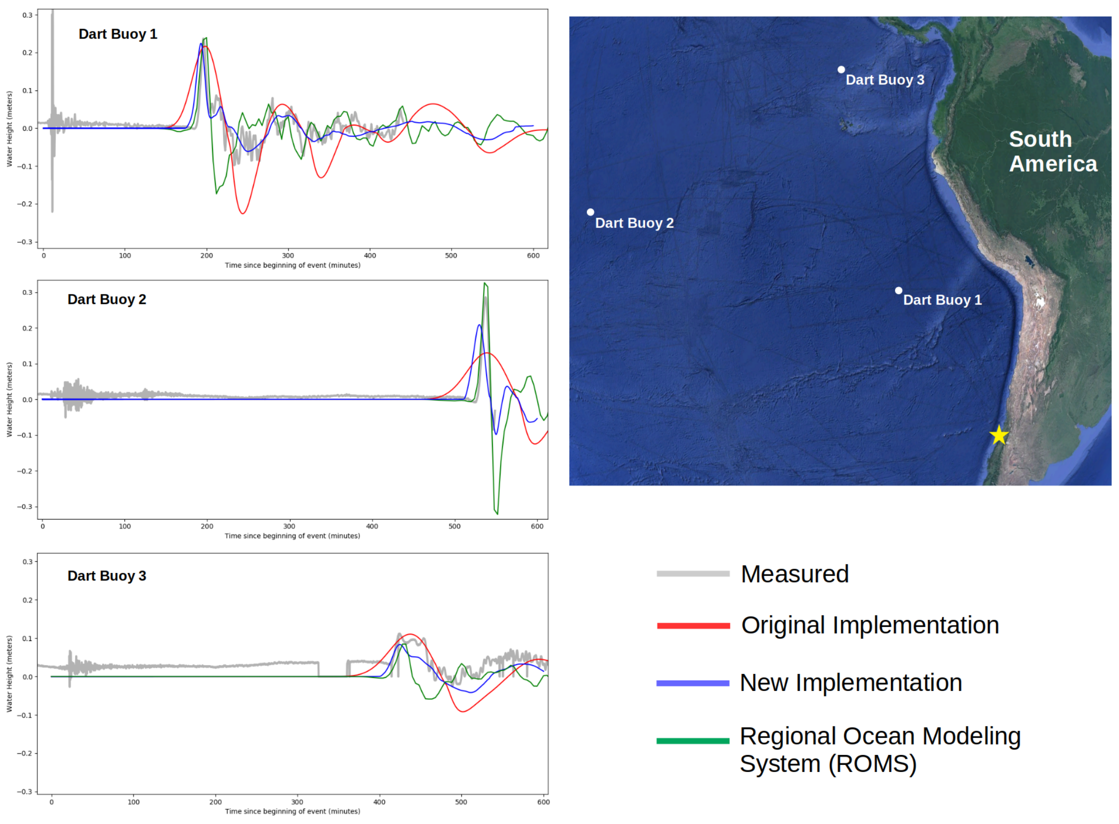


Figure 3.13: Buoy comparison plots for the 2010 Maule Tsunami. The simulation using the new implementation uses an a value of 0.25. Results show better alignment in nearly all cases.

The most important features in comparison plots such as these are the arrival time, wave height, and width. Arrival time information is important for early warning purposes, and the height and width of the initial peak are an indication of overall wave energy. Wave shape would therefore affect the complicated dynamics of water flow onto land.

Figures 3.12 and 3.13 show overall greater alignment in the aspects of wave height, width, and arrival time. Sources of error in the comparison between Tsunami Squares and ROMS for both simulations include lower resolution bathymetry near the coast which is known to affect wave dynamics (Tang et al., 2008), higher time step, and using instantaneous seafloor uplift instead of a time-dependent one.

Errors for root mean square error, initial peak height, and following trough height are shown in Figure 3.14 for the 2011 Tohoku Tsunami event. Results for the 2010 Maule event were not analyzed as there were insufficient buoys to make a meaningful comparison.

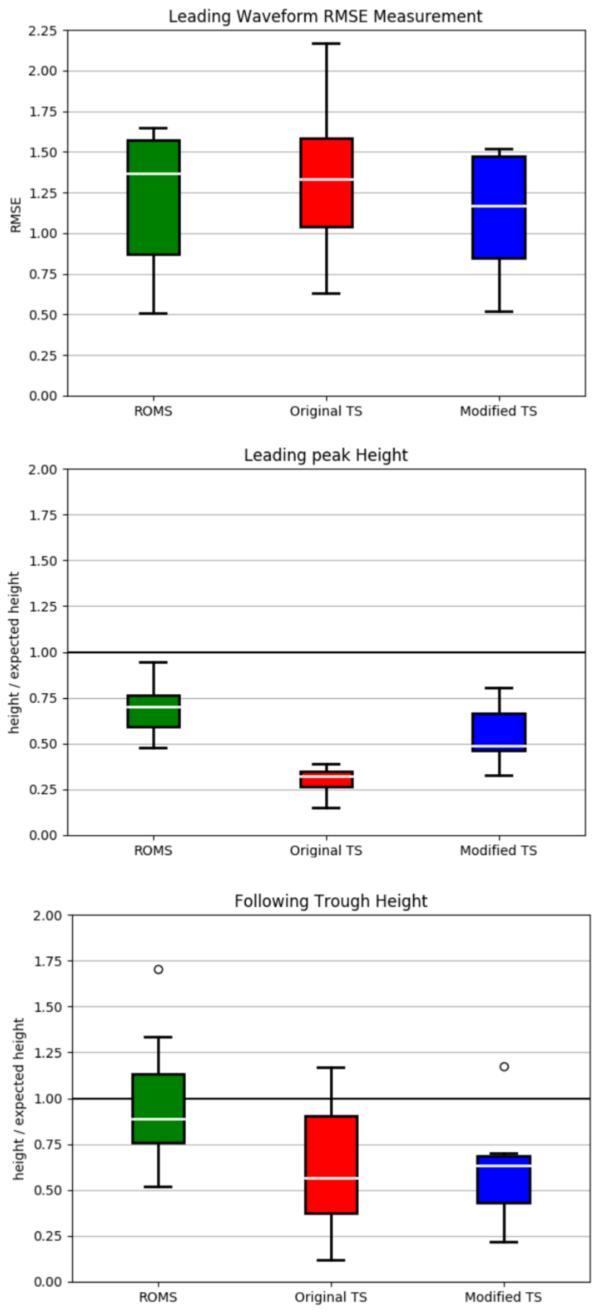


Figure 3.14: Error analysis for the 2011 Tohoku event. Left: Root mean square error using data from the start of the wave to the first trough. Middle: Ratio of the initial peak height to the height measurement by the buoy. Right: Ratio of the initial trough to the expected height measured by the buoy. Black line indicates expected result for the middle and right plots.

Leading waveform RMSE results show that the modified TS algorithm performed similarly to the ROMS simulator. The original TS algorithm, although having a similar median compared to ROMS, has a particularly high upper extreme. The leading peak height is arguably one of the most important measures when determining tsunami risk level. ROMS performs best in this measure showing a median height to expected height ratio of 0.70. The next best result is the modified algorithm with a ratio of 0.49. The original TS algorithm performs the worst with a ratio of 0.32. The trough height to expected height ratio for ROMS is very accurate with a median ratio of 0.85. The median of the modified algorithm (0.66) is higher than the median of the original algorithm (0.57). However, the original algorithm produces a larger range for this measure, meaning there will be nearly accurate results, and results that are very poor.

Sources of error in the comparison to ROMS may be due to the initial condition used. The ROMS simulation imparts some initial velocity onto the wave due to the horizontal motion of the seafloor, in addition to a time dependent uplift of the seafloor. The TS simulations did not implement these features and instead used an instantaneous vertical uplift with no horizontal velocity component. These can account for some of the difference in results seen particularly in the peak and trough heights. A purer comparison with exactly the same initial condition and simulation parameters is seen between the original and modified algorithms. In all three measures of accuracy the modified algorithm outperformed the original algorithm in terms of the median.

3.8 Other Miscellaneous Improvements

Discussed here are several smaller, yet significant, improvements made to TS. These include the following:

- Enabling simulations to cross the International Date Line
- Expanding the maximum size of the simulation
- Fixing sea to land conservation issues

The International Date Line (IDL) is a demarcation running from the North Pole to the South Pole, serving as the boundary between one calendar day and the next. It runs approximately through the center of the Pacific Ocean and is the line opposite to the Prime Meridian. The issue related to the IDL stems from a C++ package called GeographicLib, which manages points and shapes on a global coordinate system. The position of each square is therefore fundamentally connected to this library. However, the coordinate system of GeographicLib is set up such that the origin 0° longitude is positioned at the Prime Meridian, extending East to +180° and West to −180°. This is an issue because if one wants to simulate a square moving from one side of the IDL to the other, the position of the square would need to jump from +180° to −180°. To remedy this, a simple fix of shifting the entire coordinate system was made such that the longitude of the IDL was redefined as 0°. The simulation was then allowed to propagate using this shifted coordinate system without issue. This offered the most simple yet most effective approach to solving this issue. A basin-wide simulation of the 2011 Tohoku Event can be seen below, showcasing the fix.

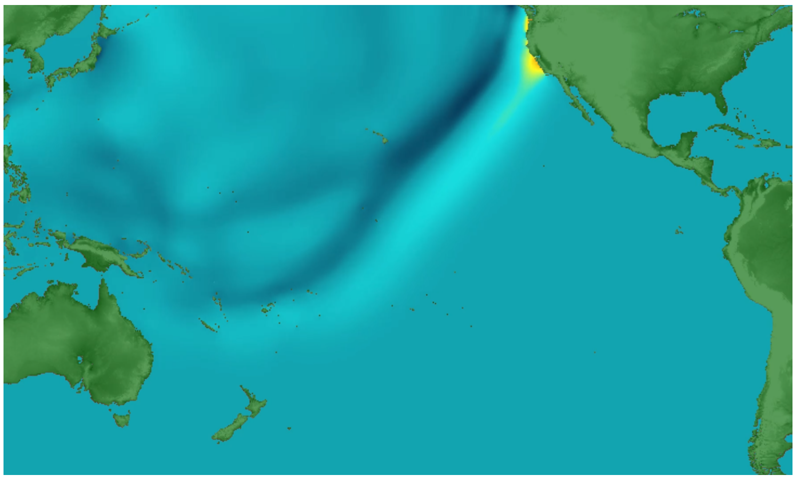


Figure 3.15: TS simulation showing the newfound capability to produce a basin wide tsunami which crosses the International Date Line. The wave is produced using the initial earthquake uplift from the 2011 Tohoku event.

The problem limiting the maximum size of the simulation is actually a problem of memory allocation. The way the code was originally written, the maximum number of squares allowed in a simulation was confined to ∼700,000 squares (a 830x830 square grid). This is very limiting in some cases, as the resolution needed to be lowered significantly for simulations spanning a large area. The problem was fixed by manually allocating the memory needed to load the bathymetry file.

The last category of error relates to conservation of momentum and mass near the coast. These include specific scenarios where a single square is left with a very small water column height during the propagation or smoothing process. For example, if a square's water column height is very small, imparting any appreciable amount of momentum to it may cause the velocity to blow up. To ensure the simulation does not have any runaway velocities or runaway volume problems, each situation was handled separately and will not be fully discussed here.

To display the results of the improvements mentioned thus far, the tsunami that struck New Zealand's North Island following the 8.1 magnitude earthquake near the Kermadec Islands in 2021 is simulated. This particular simulation showcases the improved waveform resolution.

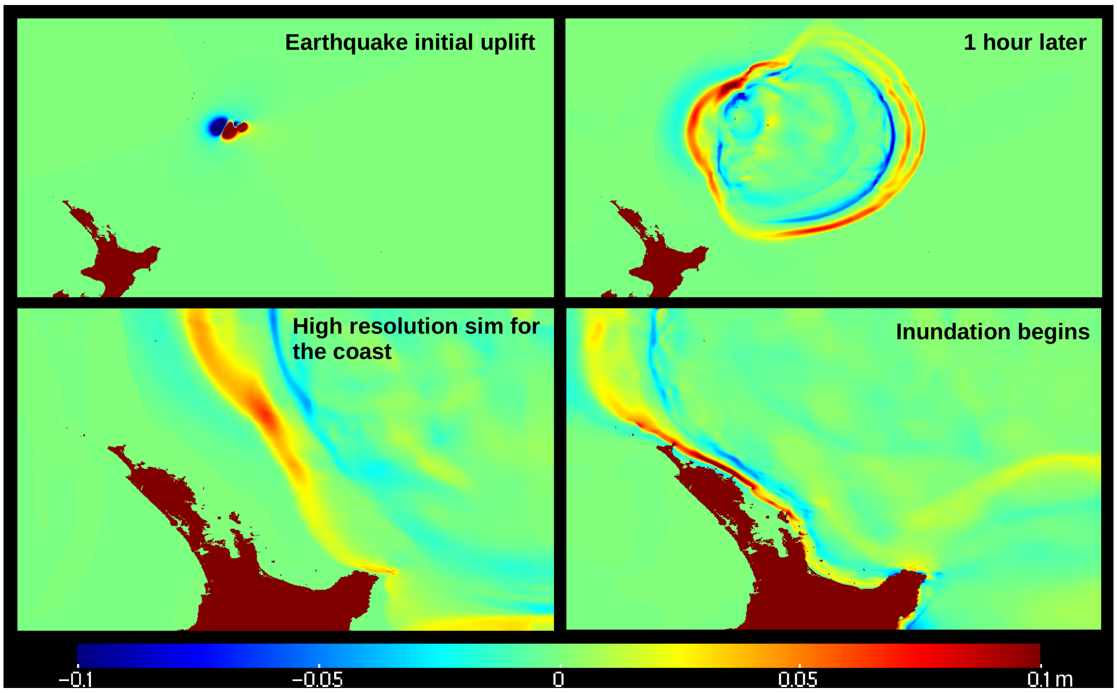


Figure 3.16: Simulation of the magnitude 8.1 earthquake that struck the coast of New Zealand in 2021. Top: Initial condition is simulated on a low resolution grid until the wave reaches the coast. Bottom: A higher resolution simulation of the New Zealand coast.

The following results show a hypothetical wave in which a 100 m tall Gaussian pile of water is simulated near the coast of San Francisco, U.S. This is mainly to show the capabilities of a larger simulation (>1,000,000 squares) along with the improved sea to land capabilities.

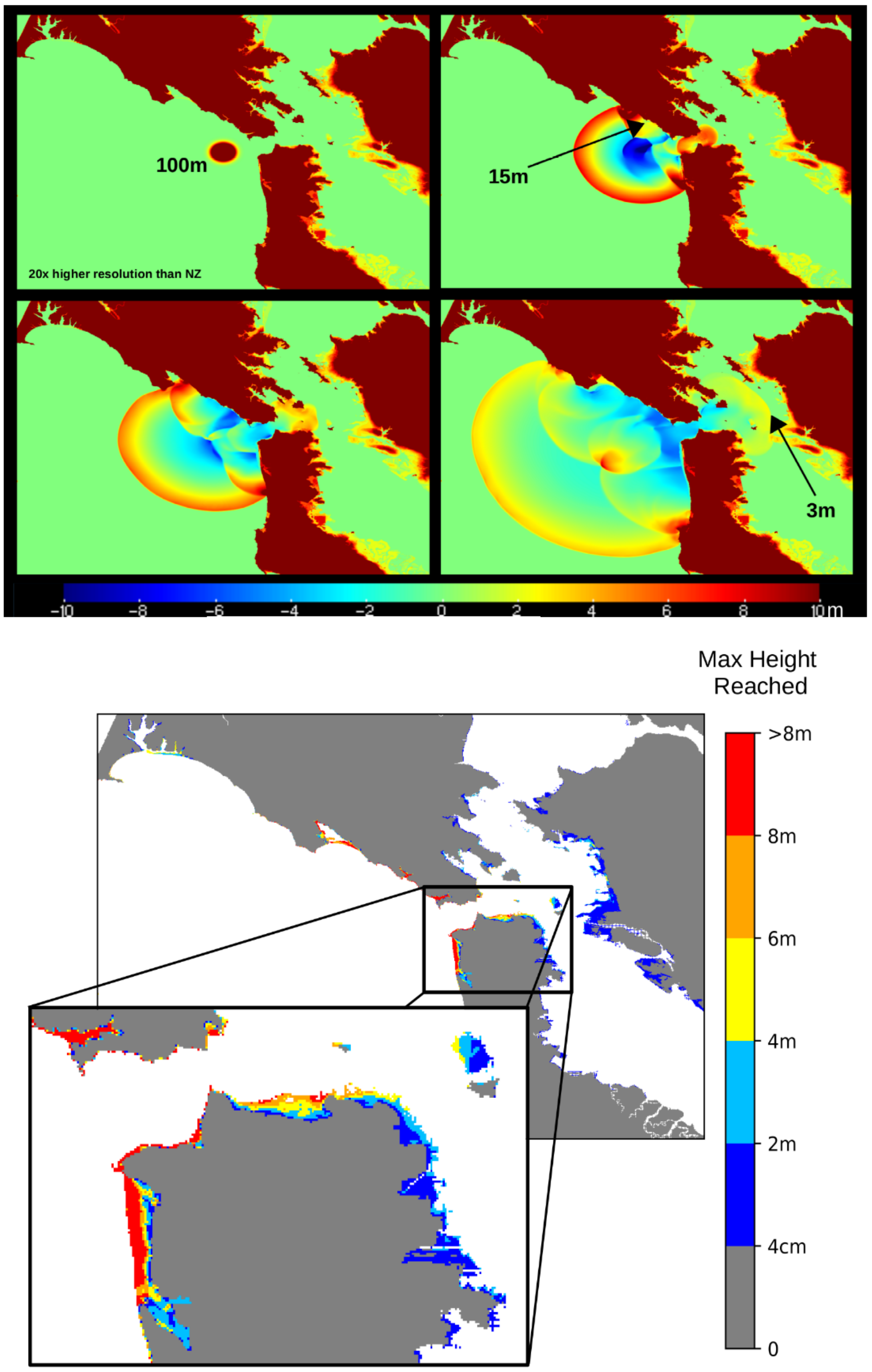


Figure 3.17: Simulation showcasing the newfound ability of TS to simulate on larger grids. Top: Initial condition of 100 m Gaussian pile of water positioned near the coast of San Francisco, U.S. Bottom: The resulting inundation map.

3.9 Conclusion

Several modifications were made to the Tsunami Squares algorithm which yield higher accuracy simulations and greater detailed waveforms. Before the changes, Tsunami

Squares could not provide enough uniqueness between simulations with similar initial conditions due to heavy smoothing, affecting the effectiveness of a searching algorithm in a given pre-computed tsunami database (Mulia et al., 2018, 2020). The modifications now leave Tsunami Squares well suited for applications which require the creation of su ch a database.

The changes made to the algorithm involve a simple change to how the slope and distance traveled are calculated for a given square, resulting in a more stable wave which requires significantly less smoothing to function. Smoothing is a added function to Tsunami Squares which takes a given waveform and eliminates any sharp features, allowing the wave to propagate with more stability. However, the more smoothing used, the more detail is lost. Thus, a reduction in the amount of smoothing used is highly beneficial for overall performance.

A new method is introduced to conserve energy. By shifting the acceleration a small distance, a sharpening effect is produced which increases the height, decreases the width, and ultimately increases the energy. Since the smoothing decreases height, increases width, and decreases energy, these two effects cancel each other out, leaving behind the original wave while smoothing over all of the small anomalies. The advantage of this method is that it does not rely on a global calculation of wave energy, meaning it conserves energy locally and allows for energy fluctuation due to complex interactions near shorelines.

Lastly, changes were made to the square-v and acceleration of squares which have opposing square-v directions. The significant reduction of smoothing uncovered errors at these locations that were previously smoothed over in the original algorithm. These errors affected the overall waveform and were detrimental to the overall accuracy of the simulation. The empirically derived fixes have now eliminated these problems.

3.10 Appendix

Table 3.2: Variable Definitions

Variable	Definition
h	height of water surface relative to sea level
H	full height of column of water from surface to floor
d	depth, measured from sea floor to sea level
square-v	velocity of an individual water column
wave-v	wave velocity ($\sqrt{gd}$)
Δx	width of a given square
Δt	length of the time step
time steps	the total number of iterations in time
t	the in-simulation time (time steps $\cdot$ Δt)
Height, σ	The height and standard deviation of a Gaussian initial condition
cells	the number of bins in the simulation
a	value from 0 to 1.0 which sharpens the wave, resulting in an increase in wave energy
f	smoothing factor, controls the smoothness of the wave, resulting in a decrease in wave energy

Chapter 4

Forecasting Tsunami Inundation using Machine Learning Techniques

4.1 Introduction

The Cascadia Subduction Zone stretches from Southern Canada down to Northern California and has a history of large, submarine earthquakes. According to records, the average recurrence rate is about 500 years with the last event occurring 300 years ago in which the entire 1000 km length of the subduction zone ruptured and produced a tsunami causing destruction reaching as far as Japan (Clague, 1997; T. H. Heaton & Hartzell, 1987; Atwater et al., 1995). It is proposed that either a large magnitude 9+ earthquake or several smaller magnitude 8+ earthquakes are estimated to occur some time in the future (T. H. Heaton & Hartzell, 1987). Because the Cascadia Subduction Zone is located only $\sim$100 km from the coast, tsunamis generated here are particularly hazardous because of how little time it takes to reach the coast. For this reason, the rapid forecasting of these tsunamis is crucial for the dissemination of warnings which can drastically reduce casualties.

In recent history, similar subduction zones have triggered large tsunamis, causing massive loss of life. The most recent example is the 2011 Tohoku tsunami, where Japan suffered a loss of life exceeding 19,500 citizens (Imamura & Anawat, 2011). In this case, the initial tsunami warning was based on an initial underestimate of the earthquake magnitude (M7.9), where the actual magnitude was much larger (M9.0) (Satake, 2014). This lead to a tsunami wave height estimate which was much lower, causing residents to stay in dangerous areas due to a misunderstanding of the risks (Ando et al., 2011). Several studies have been carried out since, aiming to produce faster and more reliable tsunami forecasts. These include the development of a system to produce real time high resolution tsunami inundation simulations using supercomputers (Musa et al., 2018), near-field tsunami forecasting based on rapid estimations of tsunami source functions (Tsushima et al., 2012, 2014), and an algorithm which finds the best matching precomputed waveforms from virtual observation points in order to directly obtain the corresponding inundation map (Gusman et al., 2014). However, the earthquake source inversions these methods rely on potentially take tens of minutes (Tsushima et al., 2011, 2012), are prone to varying results due to slightly incorrect input parameters (Mai et al., 2016), and are subject to error due to ground sensor rotation or tilt (Kubota et al., 2018). In addition, forward-modeling approaches such as the one developed by Oishi et al. (2015) was able to produce an inundation forecast in under 1.5 min, but required nearly 10,000 cores to do so and is therefore inaccessible to areas with more limited resources. From this, several challenges need to be met in order to increase the speed and accuracy of near-field tsunami inundation forecasts in the event of a Cascadia Subduction Zone rupture.

To respond to these challenges, several studies have since expanded on work done by Gusman et al. (2014) by using increasingly efficient selection techniques on precomputed tsunami databases. Mulia et al. (2018) improves on their method of finding the best fitting scenario based on waveforms from virtual observation points, and instead uses an interpolation algorithm on a map of low resolution maximum tsunami heights to produce a unique inundation map. Following this study, Mulia et al. (2020) replaces the interpolation algorithm with a two-stage deep neural network in order to gain additional efficiency. Similar to this approach, Fauzi & Mizutani (2020) uses a comparable database of low and high resolution simulations to train a convolutional neural network (CNN) and a multilayer perceptron (MLP) find the best match in the database or to produce a unique solution. In a different approach, Makinoshima et al. (2021) utilizes a CNN to forecast inundation waveforms at specific locations along the Japan coast, using wave height time series data from many buoys as input to the network.

In this study, we employ a CNN to forecast inundation resulting from a given rupture of the Cascadia Subduction Zone based on currently existing and hypothetical configurations of wave height measurement buoys. This method avoids the inherent inaccuracies and delays associated with earthquake source inversion techniques by directly using wave height time series buoy data similar to Makinoshima et al. (2021). In addition, we forecast the entire maximum inundation map as was done by Mulia et al. (2020); Fauzi & Mizutani (2020). Dissimilar to the studies mentioned, the earthquake database here is created using techniques similar to Goda et al. (2018), where correlations are introduced between slip values on the fault, creating localized asperities which heavily influence the location of affected coastal areas (Mueller et al., 2015). For this study, Seaside, Oregon is chosen as the site of interest because of its low elevation, proximity to the coast, and overall susceptibility to a future tsunami disaster (González et al., 2009; Dominey-Howes et al., 2010; Park & Cox, 2016; Park et al., 2019).

4.2 Inundation Forecasting Method

The goal of tsunami forecasting is to ultimately convert any information known about the earthquake and resulting wave into an inundation prediction before the wave reaches the shore. Typical methods use a combination of earthquake, tsunami, and inundation simulations stitched together in a real time forward modeling approach to produce an inundation prediction that can inform officials. Since this is costly in terms of the computing power and time needed for such a computation, we look for a way to obviate the need for direct computation of these simulations.

In this study, machine learning methods are employed to connect information measured soon after a tsunamigenic earthquake to a corresponding inundation map. Two types of measurements were considered as input data: global navigation satellite system

(GNSS) ground dislocation data measured at several stations along the coast and wave height data collected by buoys positioned at several locations in the open ocean. For buoys near the event, this would tell us the height of the wave relative to sea level once every 15 seconds, comprising a time series. The number of buoys would determine the number of time series that would be fed into the neural network, whereas the number of GNSS stations would determine the number of single valued data points that would be fed into the network.

Figure 4.1 separates the inundation forecasting method into two categories: precomputed and real time. In the precomputed category, thousands of earthquake, tsunami, and inundation simulations are computed to generate a database of realistic events. This is used as training data for the neural network to link the input measurements directly to inundation maps. This means after the network is trained, any supplied input data will return an inundation map within a few hundred milliseconds. In real time, if an earthquake event were to occur, the corresponding GNSS uplift data or buoy data would be fed into the neural network and return a unique inundation map. By precomputing the tsunami scenarios beforehand, tens of minutes are saved due to the elimination of a direct computation of both the tsunami and inundation simulations.

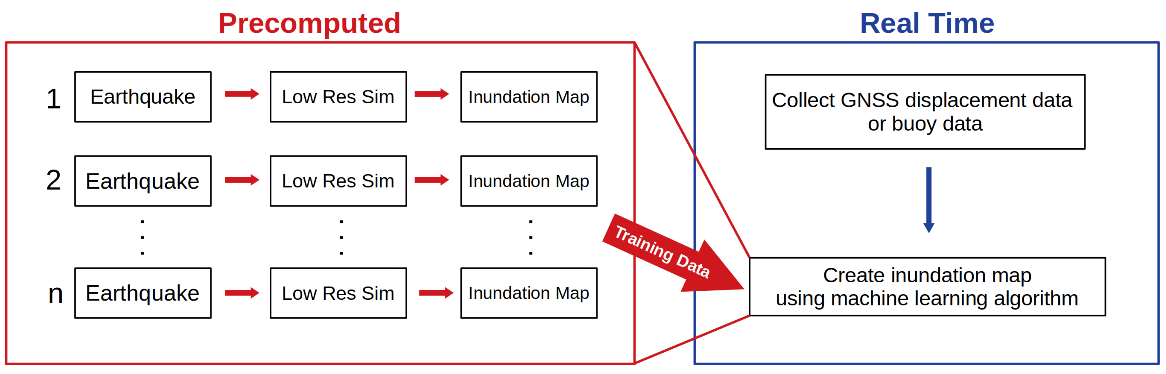


Figure 4.1: Diagram separating the precomputed and real time elements of the inundation forecasting technique. A precomputed database of simulated earthquakes, tsunamis, and inundation maps are used to train a neural network which will be able to give a fast inundation prediction.

It is important to discuss the main differences between the two types of input data that were used in this study. The two main differences reside in when the data is measured and how information-dense the data is. GNSS surface displacement data is measured at the time of rupture, whereas buoy data directly measures the tsunami waveform and can collect data for as long as necessary. This indicates that for surface displacement data, measurements can be immediately fed into the neural network for a fast inundation forecast, whereas buoys may take several minutes to collect data first before being fed into the neural network. However, the buoy data contains much more direct information about the wave since it is a clear representation of the tsunami waveform itself. Compared to displacement data at select points along the coast, buoy data should provide a less abstract link to inundation data, and thus should be more efficient in the neural network training process. Figure 4.2 illustrates the main differences between the two types of input data used.

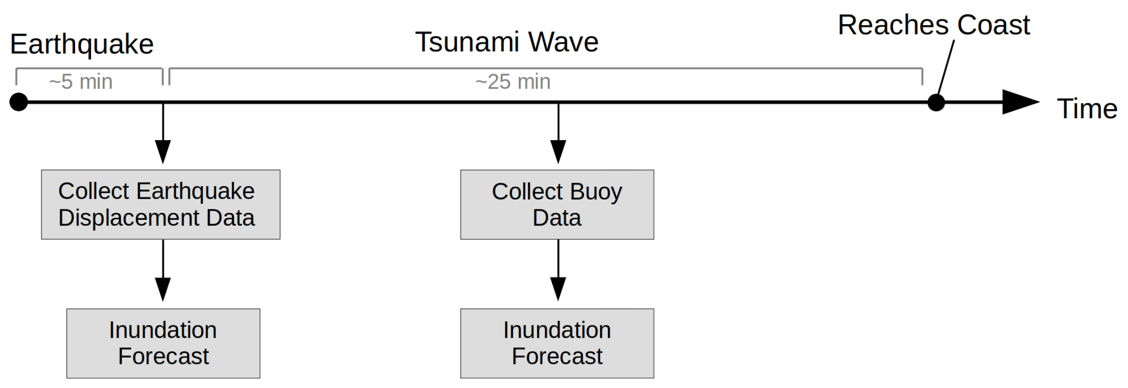


Figure 4.2: Timeline of a tsunami event in the context of the early warning methods proposed in this study. Using buoy data might contain more valuable information resulting in a more accurate prediction, but at the cost of a delay in the issuing of said prediction.

In this study, ground deformation data from simulated earthquakes and buoy data from the simulated tsunami waveforms will be used to train their own respective neural networks to explore the accuracy and feasibility of forecasting inundation. For each respective data type, the number of GNSS stations and number of buoys were modified to test how accuracy changes if more of these stations were added.

4.3 Database Development

The development of the database can be split up into three parts: the earthquake simulation, the large low resolution tsunami simulation, and the small high resolution simulation. These will be discussed individually along with a brief overview of the study site. In addition, the input data will be discussed in greater detail, describing how the data was created for use in the neural network training process.

4.3.1 Study Site

The city chosen for this study is Seaside, located on the coast of Oregon, USA. It is a relatively small city with a population of around 6,500 according to the 2010 census (Bureau, 2022). It was chosen because it is around 130 km away from the Cascadia Subduction Zone, a fault which runs parallel to the North American West Coast stretching from Vancouver to Northern California. It exists between the Juan de Fuca and North American plate, where the heavier Juan de Fuca plate slowly slides beneath the lighter North American Plate at a shallow angle. At shallower depths (<30 km), the plate is locked by frictional forces. However at deeper depths, the edge of the sinking plate becomes molten under intense heat and stress, allowing large slipping between the plates to occur, resulting in large earthquakes (magnitude 8.0+).

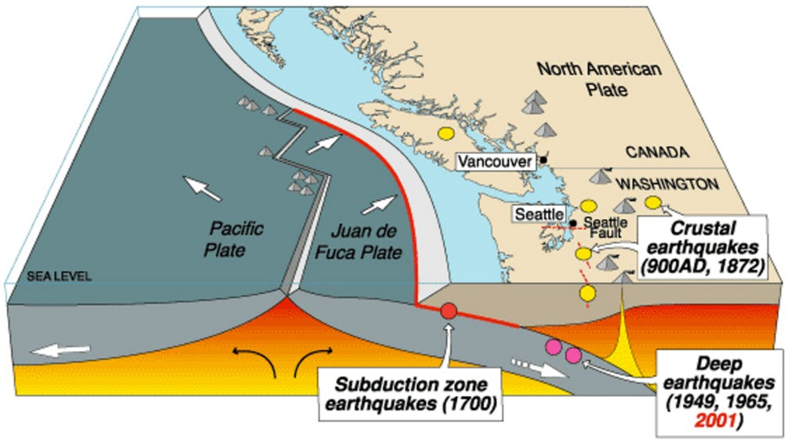


Figure 4.3: Cascadia Subduction Zone plate movement. The Juan de Fuca Plate is slowly subsiding under the North American plate, building up stress that will eventually cause large earthquakes. Courtesy U.S. Geological Survey.

Because of the massive length of this fault, it is capable of producing earthquakes that can exceed magnitude 9.0. The average interval between earthquakes is approximately 500 years, but Table 4.1 shows that the interval can vary anywhere from 910 to 210 years.

Estimated year	**Interval (years)**
1700 CE	780
780–1190 CE	210
690–730 CE	330
350–420 CE	910
660–440 BCE	400
980–890 BCE	250
1440–1340 BCE	unknown

Table 4.1: Previous known Cascadia Subduction Zone ruptures. The average interval between earthquakes is approximately 500 years. (Atwater et al., 2011)

There have been several studies detailing the probabilistic tsunami hazard assessment for this particular city (González et al., 2009; Dominey-Howes et al., 2010; Park & Cox, 2016; Park et al., 2019). In these studies, maximum tsunami amplitudes for different time spans such as 100 or 500 years are first calculated by looking at historical records and determining the largest earthquake most likely to occur in this range of time. From this, the corresponding maximum tsunami amplitude is calculated and fed into an inundation simulator to produce the inundation map. Historic studies suggest Seaside has been struck by several tsunamis in history, either from the Cascadia Subduction Zone or from other far-field tsunamis (Fiedorowicz & Peterson, 2002). It is known that once every 500 years, the Cascadia Subduction Zone ruptures, causing a massive tsunami that strikes the coast of North America (Peterson et al., 2008). However, other tsunamis which have origins elsewhere have caused flooding in Seaside. These would notably be the 1946 Alaska event, the 1960 Chile event, and the 1964 Alaska event which caused significant damage including the destruction of 12 houses and the loss of one life (Lander et al., 1993).

Seaside, Oregon is particularly vulnerable as it sits directly on a gradually sloping beach, providing little to no barrier for any large wave that comes its way. Figure 4.4 shows aerial views in addition to a map highlighting the susceptibility of the city to a large tsunami wave.

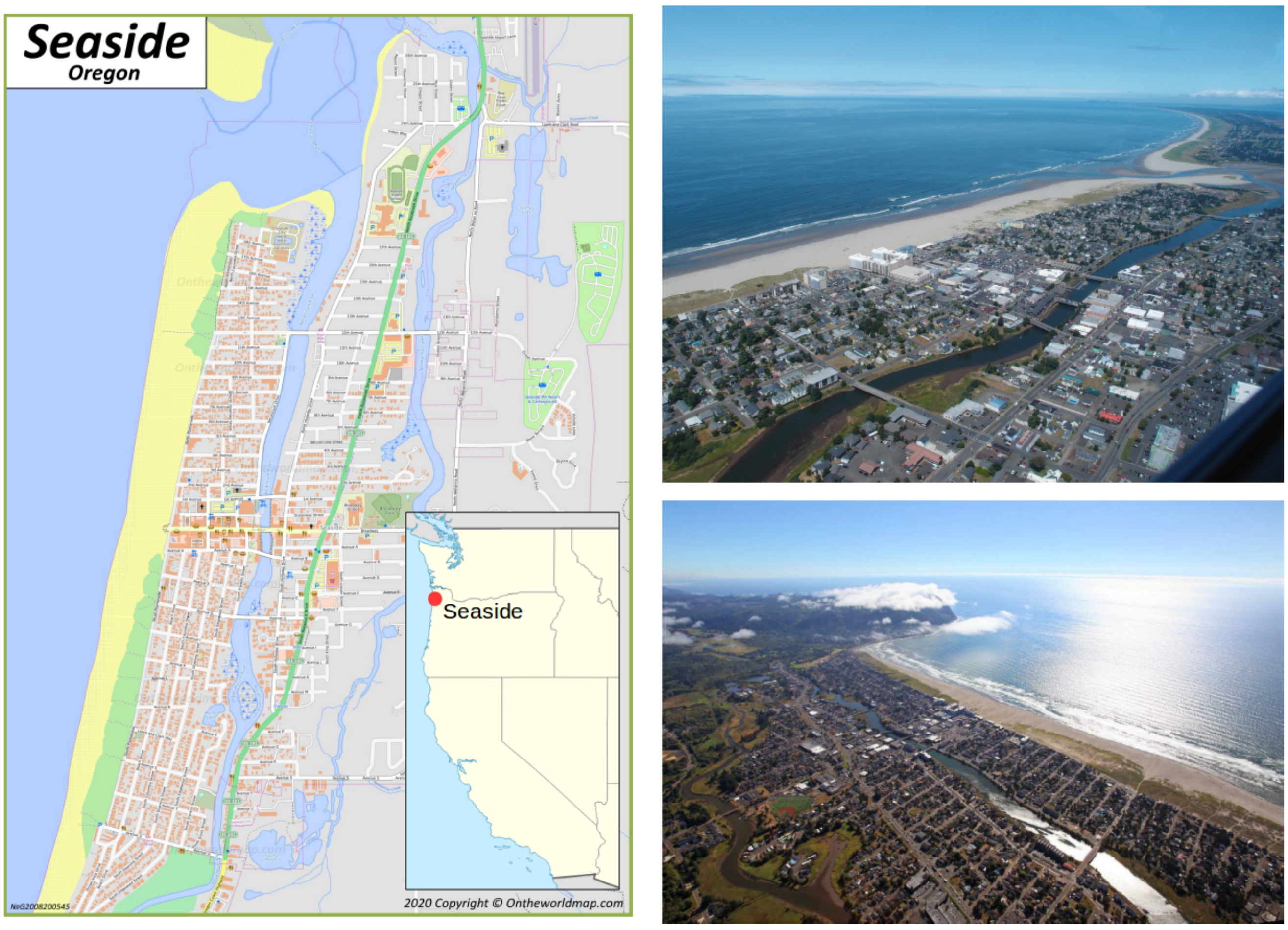


Figure 4.4: Left: Map of Seaside showing the buildings and city layout. Right: Two aerial views of the city which particularly highlights its susceptibility to an incoming tsunami.

4.3.2 Earthquake Rupture Scenarios

Generating the earthquake rupture scenarios for a tsunami database is crucial because it forms the initial conditions from which all subsequent simulations begin. If the Cascadia fault ruptures in a way which is contained within the basis defined by this database, then the neural network should be able to make an accurate prediction for inundation. In an attempt to span the set of all possible earthquakes caused by the Cascadia Subduction Zone within the Oregon region, a database of 3000 earthquakes was made.

Typically, simulated faults are defined by a rectangular plane extending into the earth at a precise angle. This is also how the Cascadia fault is modeled in this study. These faults are first defined by specifying its location, length, width, depth, rake, strike, and dip. These quantities precisely describe the position and orientation of a rectangular plane located beneath the Earth's surface. The length and width describe the size of the rectangular plane, the location and depth describe the position of one of the two upper-most corners, the rake angle and dip angle control the rotation of the plane, and the strike angle controls the slip direction. The rectangular plane is thought of as the area over which the earth slips past each other, while keeping the edges of the plane fixed. This essentially describes, in the most simplest form, what occurs during an earthquake.

After the fault is defined, it is often partitioned into a number of subfaults or sections such that each have their own slip distance and direction. Once the slip parameters have been chosen for each subfault, the 3-dimensional deformation of the seafloor can be calculated using analytic formulas for the dislocation of an elastic half-space due to shear stress (Okada, 1985). Typically, the z-direction component is translated directly to the water surface to obtain the initial condition for a tsunami simulation.

When developing such a collection of earthquakes, each one must remain sufficiently unique such that there are no substantially overlapping scenarios in the database. To accomplish this, previous studies involving machine learning on a database of tsunamis either take each subfault and randomly assign a slip value (Makinoshima et al., 2021), or systematically vary the position and size of the fault over a region (Mulia et al., 2020; Fauzi & Mizutani, 2020). However in this study, more sophisticated methods were used.

The procedure for generating a database of realistic earthquakes was inspired by Goda et al. (2018). There, they created earthquake rupture models for the Nankai-Tonankai subduction zone near Japan based on the measured characteristics of the 2011 Tohoku earthquake. They created of wide variety of realistic slip distributions in order to study probabilistic tsunami hazard and risk assessment. In order to create their slip distributions, a number of parameters were chosen such as the Hurst exponent to control the correlation length, the Box-Cox power parameter, mean and maximum target slips, nucleation point, and rupture velocity. Particular attention was given to the correlation length and the target maximum target slips which were used to create the collection of stochastic slip distributions used in this study.

Shown in Figure 4.5 are five stochastically generated slip distributions created by a process similar to Goda et al. (2018).

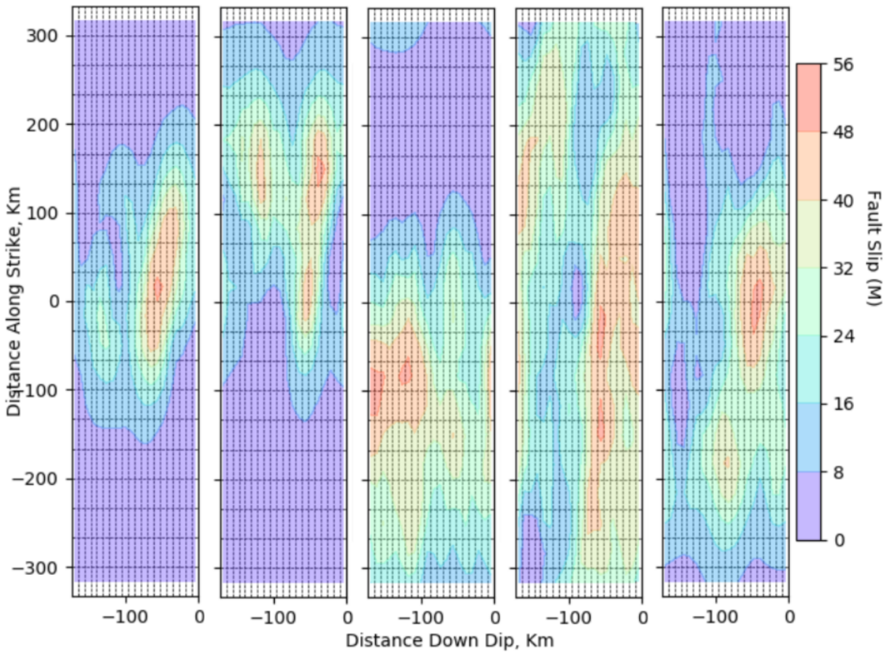


Figure 4.5: Five randomly generated slip distributions used in this study made by introducing correlations, setting maximum slip targets, and applying cutoffs.

The procedure used to create slip distributions such as the ones depicted in Figure 4.5 begins with the definition of the rectangular fault itself. The rectangular fault defined here stretches ∼660 km from Northern Washington to Southern Oregon with a width of about ∼170 km. The upper most edge of the fault follows the Cascadia fault line and extends downward at a shallow angle of ten degrees towards the East where it reaches a depth of 30 km. The fault is split up into a 20x20 grid of rectangular subfaults, totaling 400 subfaults. Each rectangular subfault is given its own individual slip value, but the direction of the slipping is unidirectional.

Once the fault is defined, slip values can be assigned to each of the 400 subfaults. First, each subfault is given a value from a Gaussian distribution of mean 10 m and standard deviation 60 m. Negative slip values will be generated from this distribution but they will be handled later. Then, a portion of the fault is chosen along the latitudinal direction. The slip values within this range are then multiplied by an arbitrarily large factor such as 20 as used here. This is to prioritize a randomly selected region of the fault, as one cannot assume the entire defined fault will rupture at once. For example, in Figure 4.5, the left most slip distribution was generated by amplifying the slip values from -150 km to 200 km.

After the rupture is localized, correlations are introduced to the 20x20 grid of subfaults by way of the Fourier filter technique, which takes a grid of random values and transforms them into the frequency domain where the slope of the power spectral decay is manually set by choosing the Hurst number. This technique has also been implemented by Løvholt et al. (2012) where 500 heterogeneous slip realizations were generated for an analysis on he variability of runup.

The process of introducing correlations is as follows. After the grid has been initialized by some set of randomly generated values h_{ij}, a fast Fourier transform of the 2D grid is taken, giving H_{ij}. The radial wave number k_{ij} is first defined as

$$k_{ij} = \sqrt{1 + i^2 + j^2} \tag{4.1}$$

Then, the Fourier transformed values H_{ij} are scaled by factor $k_{ij}^{\beta/2}$

$$H'_{ij} = H_{ij}/k_{ij}^{\beta/2} \tag{4.2}$$

where β is defined according to the Hurst parameter H.

$$\beta = 2(H + 1) \tag{4.3}$$

The Hurst parameter is set to one in this study, indicating maximal correlation for this grid.

An inverse fast Fourier transform is then applied to obtain the new correlated slip values h'_{ij}. Afterwards, as done in Goda et al. (2018), the current slip distribution is scaled such that its peak reaches a target maximum slip of 50 m. The negative slip values are then set to 0 to prevent slip in the opposite direction. The resulting slip map can be seen in Figure 4.5.

In order to obtain the surface displacement of the seafloor due to the slipping of the rectangular subfaults, analytical solutions of dislocation in an elastic half-space are used (Okada, 1985). However here, only the vertical component of displacement is considered. Figure 4.6 shows an example of the final slip distribution along with the corresponding vertical component of seafloor displacement used as the initial condition for a tsunami simulation.

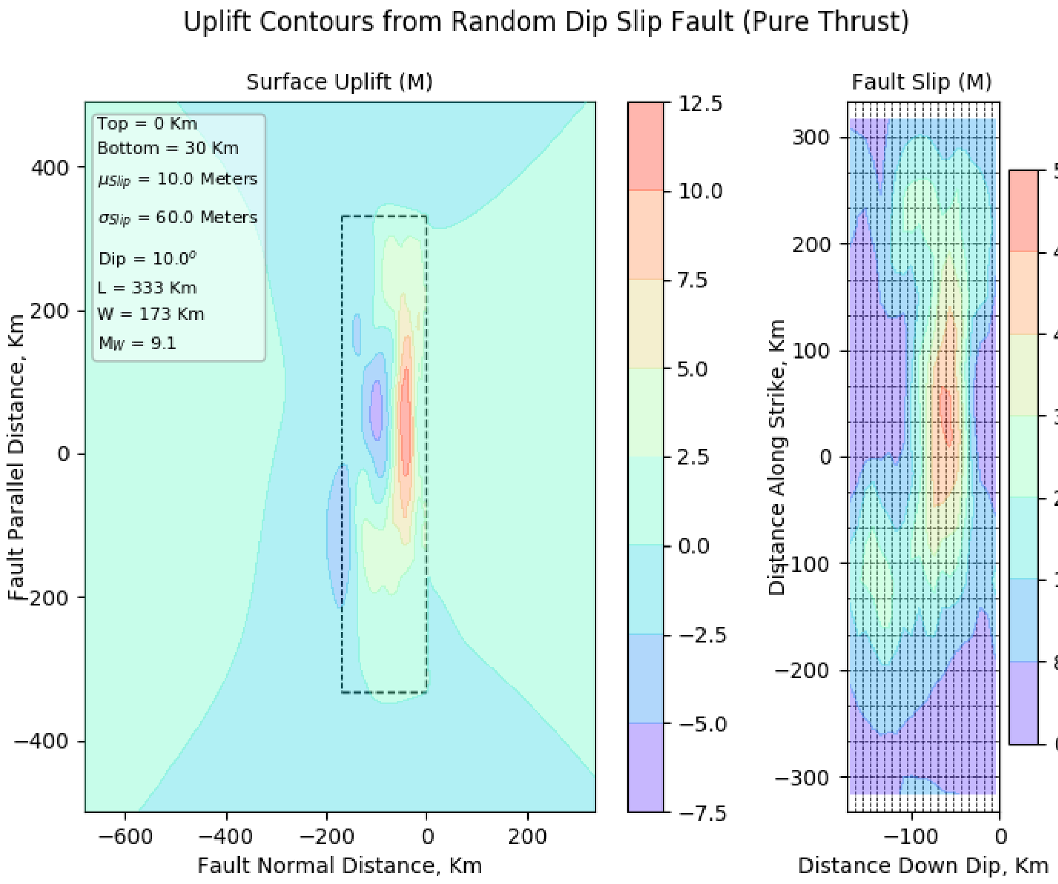


Figure 4.6: Right: A randomly generated slip distribution generated by introducing correlations, setting maximum slip targets, and applying cutoffs. Left: The resulting vertical displacement on the surface of the earth.

The moment magnitude for each of the 3000 earthquakes in the database were calculated. The calculation starts with the definition of the seismic moment M_0 which is expressed by

$$M_0 = DA\mu \tag{4.4}$$

where D is the average displacement of the faults, A is the area over which the rupture occurred, and μ is the rigidity of the rock around the fault and is expressed in units of newton-meters. This is essentially the energy released by an earthquake.

The seismic moment M_0 is then used to calculate the moment magnitude M_W to allow for a more uniform scale by which to compare to other earthquakes. It is defined as

$$M_W = 2/3 \log_{10} M_0 - 10.7 \tag{4.5}$$

The distribution of the moment magnitudes M_W produced by the procedure above for all 3000 earthquakes can be seen in Figure 4.7.

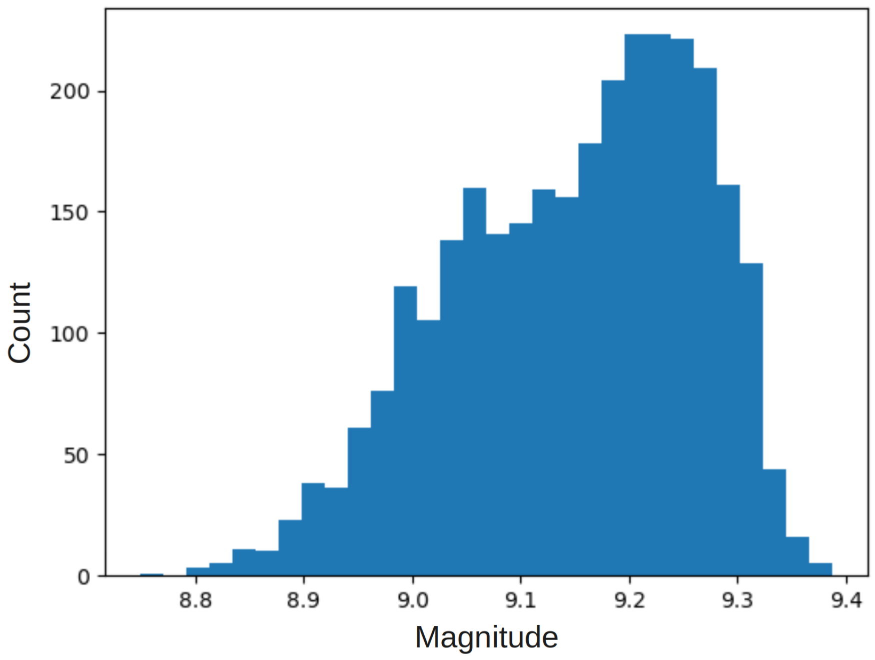


Figure 4.7: Distribution of magnitudes for all 3000 earthquakes in the earthquake database. Magnitudes range from ~8.8 to ~9.4.

Since the mean slip values were not adjusted to be log-normal, the distribution of magnitudes is not expected to be normal. The magnitudes range from 8.8 to nearly 9.4, representing a large Cascadia Subduction Zone rupture. The machine learning algorithm's predictability will therefore be restricted to earthquakes of magnitudes within this range.

4.3.3 Tsunami and Inundation Simulations

The tsunami and inundation simulations were both carried out by Tsunami Squares (TS), a wave and inundation simulator which uses a cellular automata approach to equivalently solve non-linear wave equations, making it computationally efficient. As mentioned in previous chapters, it has been tested against numerous historic tsunamis and landslides as well as other proven simulators to verify its accuracy (Xiao et al., 2015; Wang et al., 2019; Wilson et al., 2020). In addition, the recent improvements made to TS increase the resolution of the waveform, allowing more information to be preserved from the initial condition. For the purposes of creating a database of tsunami and inundation simulations, TS is a highly suitable as it is computationally fast and now has sufficient resolution to produce distinguishable tsunami simulations starting from similar initial conditions. Distinguishability is very important in the creation of such a database because it ensures every unique earthquake event is paired uniquely with its own unique tsunami, decreasing the number of overlapping events and increasing the information contained in the database (Mulia et al., 2018, 2020). Computational speed is also important in the creation of a database as it increases the number of total simulations which can be used to train the neural network. For this study, the computing power was limited to a single Intel Core i7-8700 CPU (3.20GHz) node with 12 cores, making computational efficiency a necessity. Similar studies involving a database of inundation simulations are usually limited to a few hundred events (Mulia et al., 2020; Fauzi & Mizutani, 2020), as inundation simulations can be very costly to simulate since they rely on solving non-linear equations in order to propagate waves.

The wave simulation is split up into two separate simulations: a large, low resolution simulation of the entire region and a small, high resolution simulation of the coast. It is typical to first simulate the tsunami wave in a low resolution bathymetry grid, then switch to a high resolution bathymetry grid for the coast where small details in the topography are more important for runup. This is to save considerable computation time and resources, as using high resolution bathymetry for the entire tsunami event would be impractical.

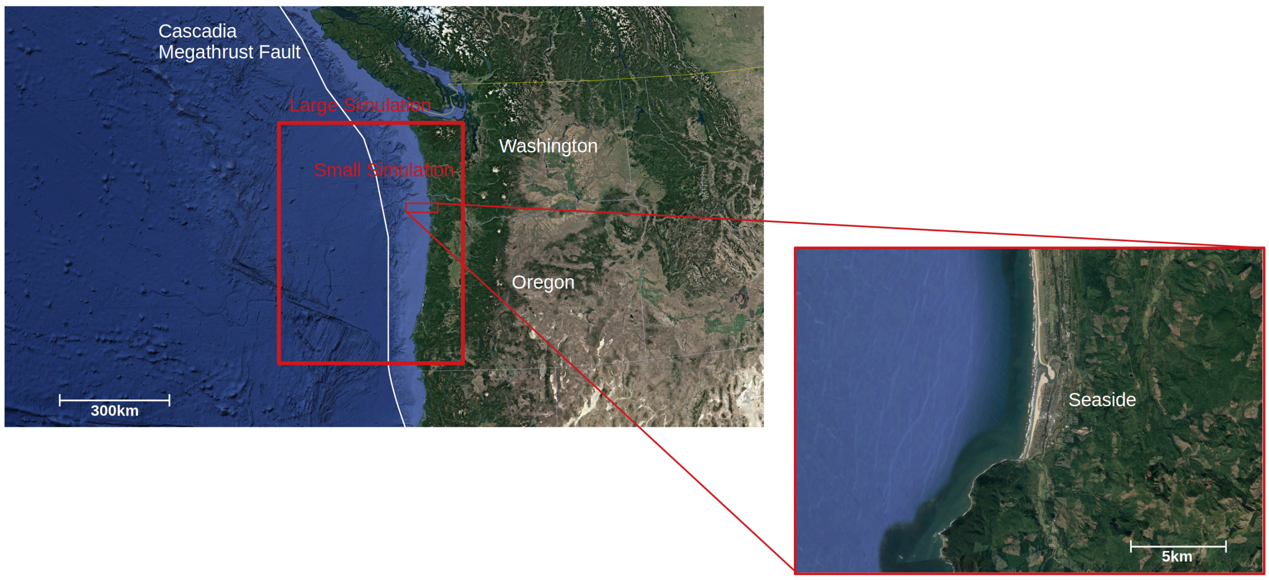


Figure 4.8: Map of the Cascadia Subduction Fault showing the simulated regions. A large, low resolution simulation is used to simulate the tsunami wave until it nears the coast. A smaller, high resolution simulation is then used for inundation.

Figure 4.8 shows how the Cascadia Subduction Zone is partitioned for the purpose of simulating. The larger simulation nearly spans the length of the Washington and Oregon coast and extends ∼500 km into the sea. Since it is so large, a 60 arc-second resolution bathymetry was used (N. N. G. D. Center, 2009). Also, to minimize computation time, a time step of four seconds was chosen which is the largest time step TS can allow given the constraints of the simulator. The wave was allowed to propagate for 200 time steps (13 minutes) at which point most waves are ∼50 km away from the coast. At this point, the last time step of the large simulation is used as the initial condition for the smaller simulation of the coast. For this, a high resolution bathymetry of the U.S. Northwest coast was used (N. N. G. D. Center, 2003). It is at a considerably higher resolution of three arc-seconds compared to the 60 arc-seconds of the larger simulation and provides enough resolution for hazard assessment along the coast. The bounds of this simulation were carefully chosen considering three criteria: ability to capture incoming waveform, ability to capture runup onto land, and size minimization. The first criteria is arguably the most important since to capture the incoming waveform, the main leading wave of the larger simulation needs to overlap with the smaller simulation at the time the large simulation ends. Therefore, the small simulation needs to extend far enough into the ocean to capture all possible waves produced by the large simulation, while also remaining short enough to conserve computational resources. The second criteria is fairly straight forward as a sufficiently large wave hitting the coast determines the maximum distance the wave reaches inland. As for the parameters of the coastal simulation, 1000 time steps (50 min) were simulated with a time step of three seconds, providing enough time for a given wave to reach the shore and drain back into the ocean. For both the low resolution and high resolution simulations, the computation time required to produce a single simulation was 20 minutes.

Figure 4.9 shows an example tsunami in the database propagating for 13 minutes until switching to the smaller simulation where a more precise simulation of inundation is carried out.

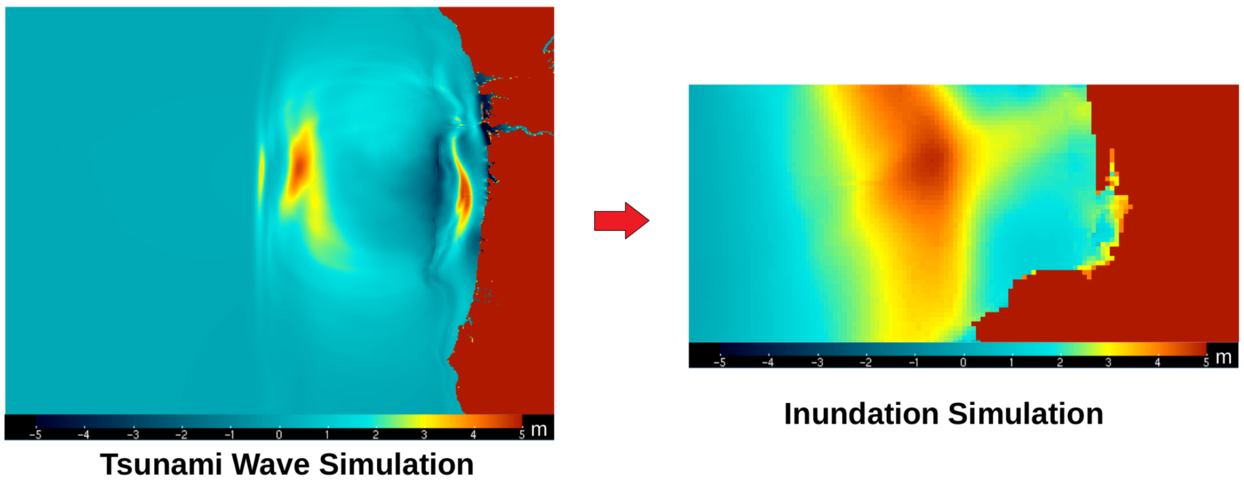


Figure 4.9: Left: Large, low resolution simulation which takes earthquake uplift as an initial condition and simulates the wave until it is near the coast. Right: Small, high resolution simulation which starts where the previous simulation left off, simulating the coastal inundation for Seaside, OR.

Figure 4.10 shows sequential frames for one of the high resolution inundation simulations in the database. As shown in this example, inundation begins as soon as ∼25 minutes after the earthquake occurs which is followed by the main wave, striking ∼10 minutes later. An analysis of the database-average waveform confirms Figure 4.10 is a representative case as most tsunamis inundate the coast ∼25 minutes after the initial earthquake. Since the time between the earthquake and inundation is so short, the 13 minutes of buoy data collection could be shortened and the accuracy could be reevaluated to provide an earlier forecast.

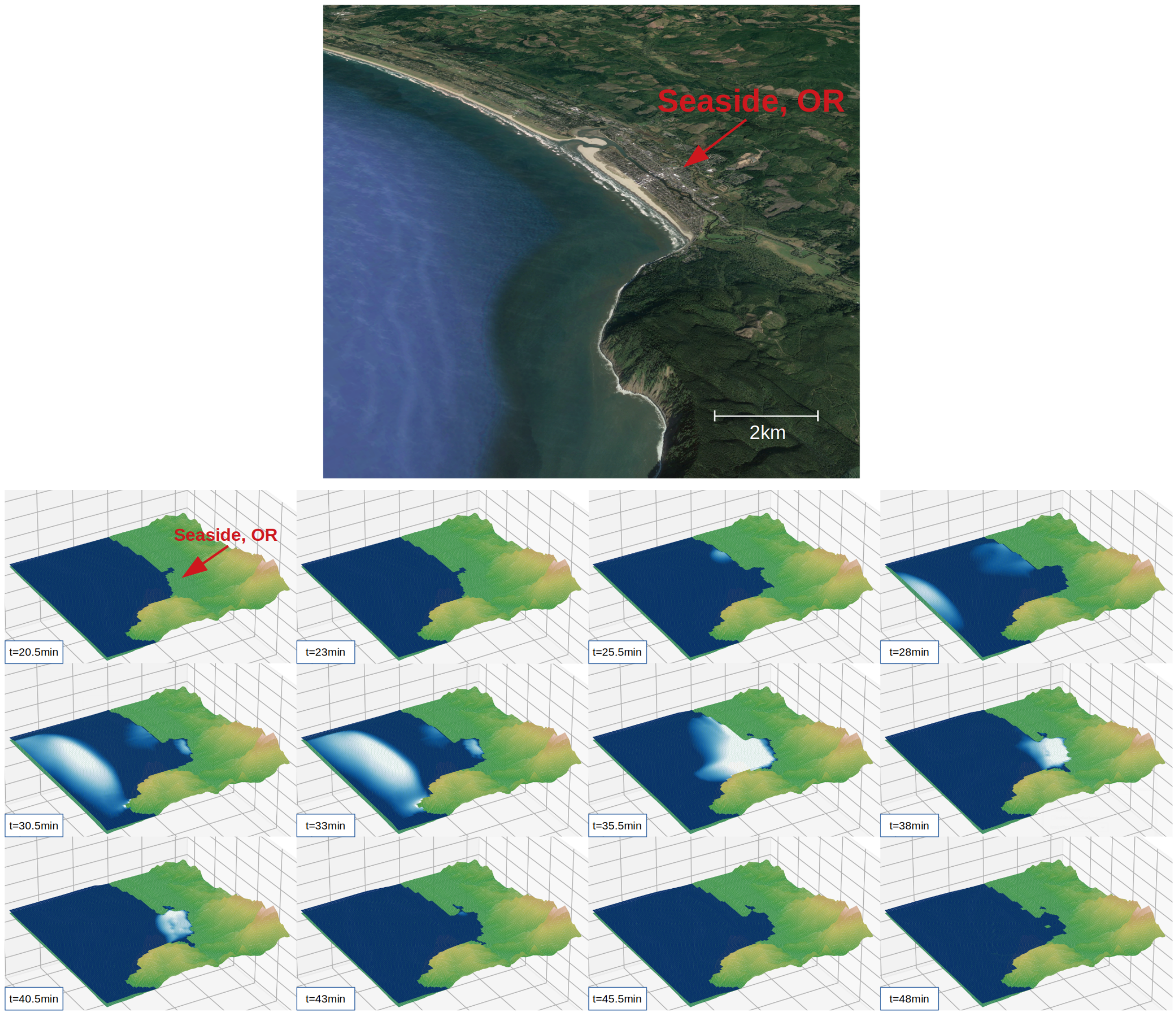


Figure 4.10: Simulation of the coast of Seaside, OR, USA for a randomly selected event in the database. Inundation begins ∼25 min after the earthquake has occurred, whereas the main wave strikes 10 min later.

Once the database of the high resolution simulations of the coast have been established, a map of the maximum flow depth is created for each simulation. Here, the flow depth is defined as the height of the water measured from the surface of the ground, not to be confused with runup height, which is measured from sea level.

The left most image in Figure 4.11 displays a inundation map resulting from a particularly large wave, showing the full extent of how far water can reach on land. If the whole map was used as input for the machine learning algorithm, the majority of the data would consist of zeros, as water never reaches the higher altitude regions and inundation is undefined in the ocean. Therefore, two cuts were made to exclude pixels with altitude greater than 33 m and pixels in the ocean. With these cuts, all important data is preserved and the amount of data points used as input are reduced from 3968 to 732. This is crucial as it improves the training efficiency and overall predictive power of the neural network by avoiding the allocation of resources to data containing no information. The cuts are shown in Figure 4.11.

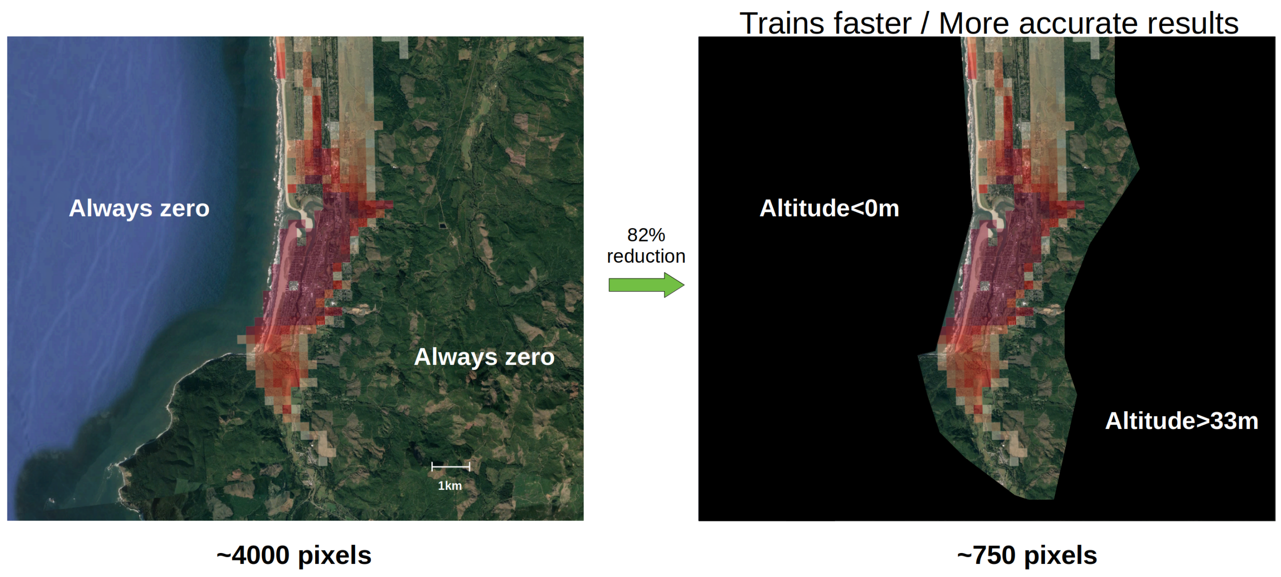


Figure 4.11: Cuts applied to the inundation map allowing the machine learning algorithm to train faster and produce more accurate results. If the altitude is less than 0 m or greater than 33 m for a given pixel, it is cut out as it is always zero for every event in the database.

Then, each of the 3000 inundation maps are flattened into a 1D arrays and written to a text file where each row represents one event. This will serve as the output of the neural networks used in this study.

4.3.4 Ground Deformation Data

Two types of input data are used in this study: ground deformation data and buoy data. They will be treated separately with their own respective neural networks and data analysis. In this section we will focus on ground deformation data and will describe the data collection instruments, the existing coverage near the Oregon and Washington region, and how it will be used as input in the neural network.

GNSS stations are designed to continuously measure its position in 3-dimensional space. They are largely positioned and used for the purpose of detecting and recording the motion caused by seismic activity among other uses such as subsidence monitoring. Depending on the type of apparatus used and mode of operation, data collection rates vary from once per second to once per 30 seconds. Positional accuracy nears a few centimeters both horizontally and vertically. Figure 4.12 shows one such station located in Port Angeles, WA.

Figure 4.12: GNSS receiver with a sampling rate of one second stationed in Port Angles, WA. (Team, 2013)

The National Oceanic and Atmospheric Administration (NOAA) manages a network of these stations known as the NOAA Continuously Operating Reference Stations Network (NCN). The NCN provides GNSS data supporting positioning, meteorology, and other geophysical applications throughout the United States (Survey, 2022). The stations near the site of interest are shown in Figure 4.13.

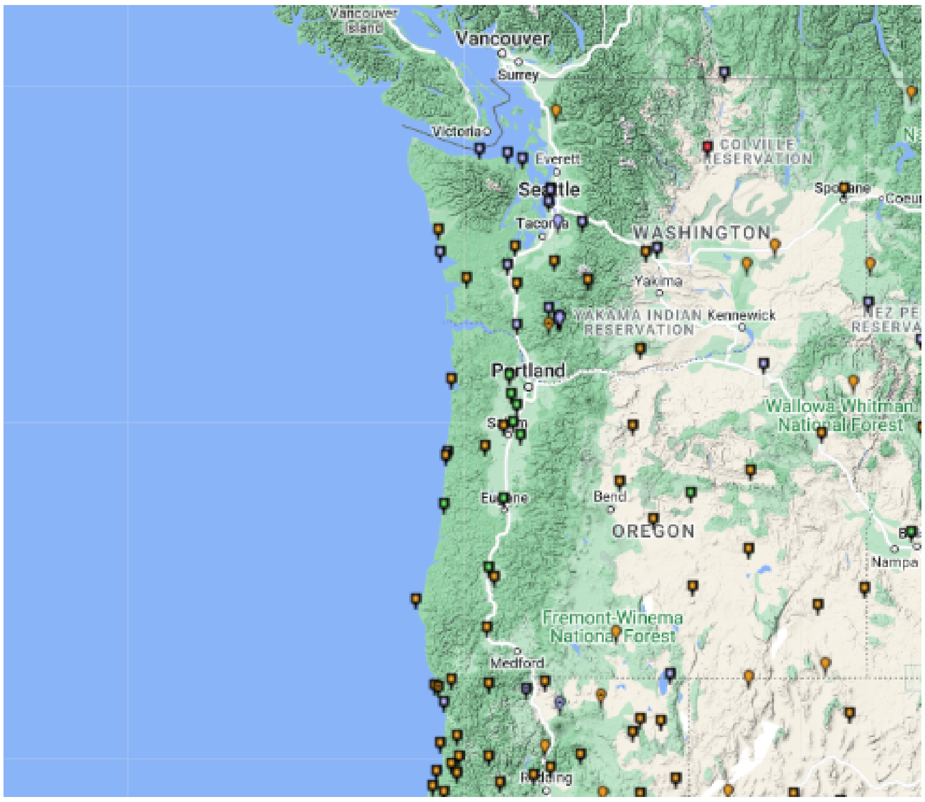


Figure 4.13: Map of currently operating GNSS observation stations for the U.S. Northwest. 29 of these stations were used as input for the neural network in order to predict inundation. (Survey, 2022)

For this study, 29 stations nearest the coast were selected. After a given earthquake, the vertical displacement of these stations will be measured and recorded. These values would then be arranged into a 1D array and fed into the neural network where it would produce an inundation prediction. Virtual ground deformation measurements at each of the 29 points are easily obtained since the earthquake simulation already outputs a 2D array of vertical displacement. The 3000 1D arrays are then written to a text file which comprise the input for the neural network.

Other hypothetical configurations of GNSS stations were also tested. In total, nine other configurations were tested, ranging from 500 observation points to 4500 observation points. These configurations include GNSS stations which are positioned in the open water, which although unrealistic, would provide valuable information as to what is needed in order to form an accurate prediction. The aim for these tests were to determine the predictive power of this method, while not being limited to currently available stations.

4.3.5 Buoy Data

Tsunami detection buoys are designed to measure tsunami wave heights in the open ocean generated by underwater earthquakes. They are deployed specifically for the purpose of early detection of tsunamis to aid in the issuing of an accurate warning. Displayed in Figure 4.14 is one such buoy off the coast of Tillamook, Oregon.

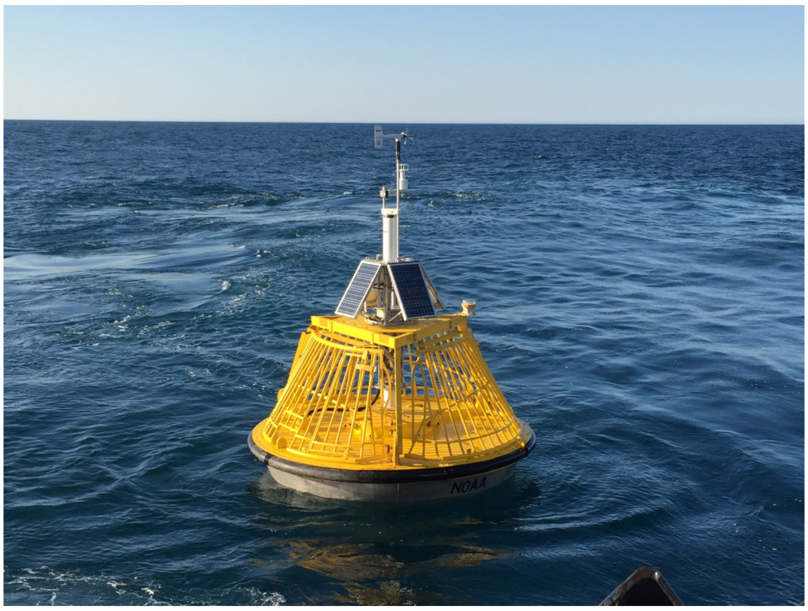


Figure 4.14: Waveheight measurement buoy located off the coast of Tillamook, OR. The pressure sensor gauge responsible for recording the wave height is attached to the buoy below. The buoy itself is responsible for the transmission of data. (N. N. D. B. Center, 2022)

However, the buoy itself floating on the water does not contain the apparatus that measures the wave height. The tsunami buoy system is actually comprised of two components: a pressure sensor attached to the sea floor, and the buoy itself. The pressure sensor measures the change in the water height directly above it by taking advantage of the direct relationship between pressure and depth ($p = \rho g h$), while the buoy receives this data via acoustic telemetry and then relays it to a satellite back to a tsunami warning center where it is recorded and uploaded for public access. The buoy system operates in two modes: standard and event. In its standard mode of operation, data is recorded every 15 minutes. This is how these buoys operate the vast majority of the time. However, when a large seismic event occurs, the buoys switch into event mode where data is recorded every 15 seconds. The data used for input in the neural network assumes event mode.

The buoy data time series itself looks something like what can be seen in Figure 4.15. In this example, four randomly selected points were chosen to represent buoys in the open ocean. For a given tsunami in the database, each buoy can be represented by a time series consisting of 50 data points, each separated by 15 seconds totaling ∼13 minutes. Each buoy's time series is then stacked to form the input that the neural network will see. More on the data formatting will be discussed in further sections.

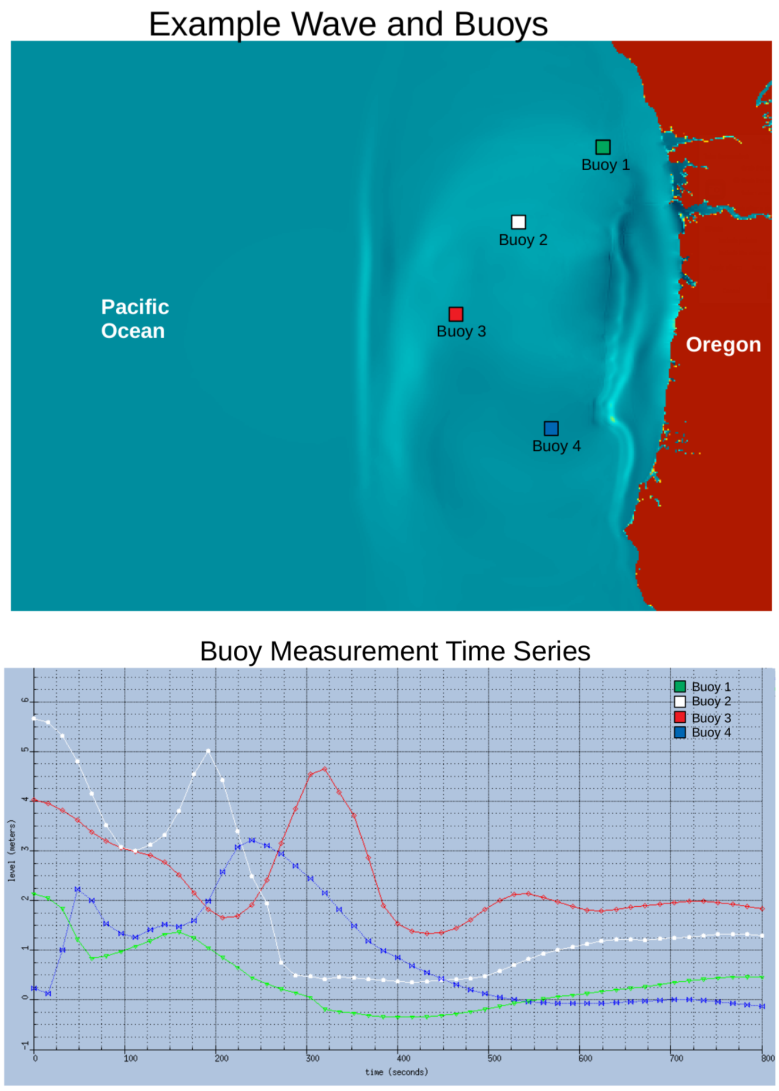


Figure 4.15: Example of simulated buoy data for four randomly chosen locations. Top: four randomly selected buoy locations for a given tsunami event in the database. Bottom: The resulting wave height time series for each of the four points.

NOAA's National Data Buoy Center has a map of all the buoys currently in service for the West Coast of the United States. These locations were referenced to test how well a neural network could forecast inundation based on existing infrastructure. The 14 buoys which were referenced are shown in Figure 4.16. These are not all of the available buoys in the simulated area. Some were removed based on their distance from the site of interest. In addition, any buoys which were very close to each other were seen as duplicates and similarly omitted.

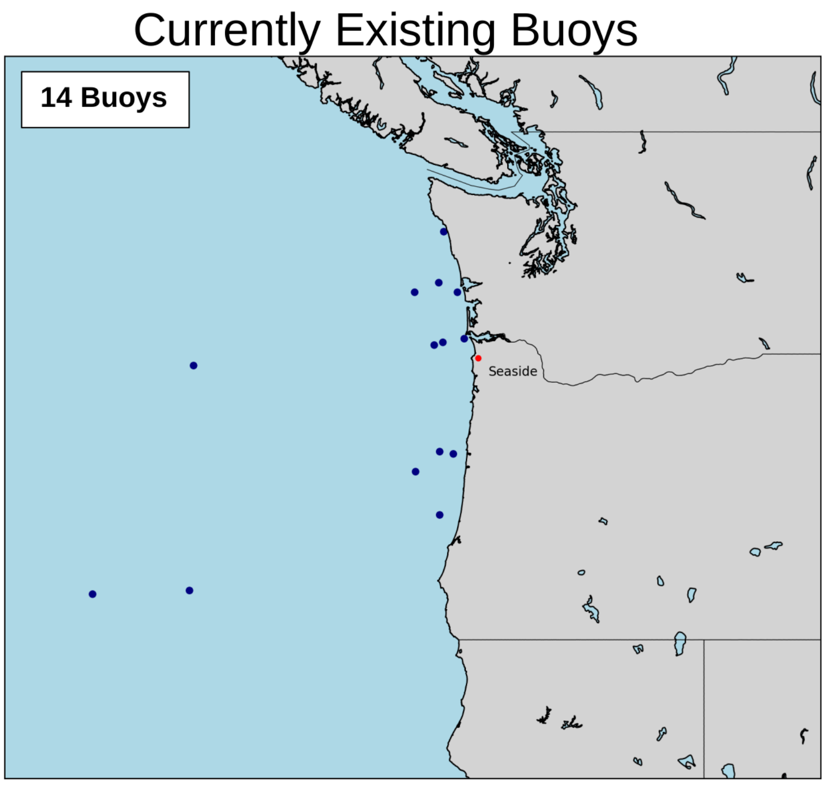


Figure 4.16: Currently operational wave height measurement buoys near the site of interest, Seaside, Oregon. (N. N. D. B. Center, 2022)

Similar to the GNSS ground deformation data, different numbers of hypothetical sensors in various configurations were tested. Grids of various separation distances were tested to explore how inundation prediction error changes with increasingly sparser grids. The spacing of the tightest grid configuration was chosen to be 18 km, with every subsequent grid expanding by 9 km. In total, 18 grid spacings were tested ranging from 18 km to 173 km with the total number of buoys ranging from ∼10 to ∼800. Figure 4.17 shows four examples of differently spaced grids used in this study.

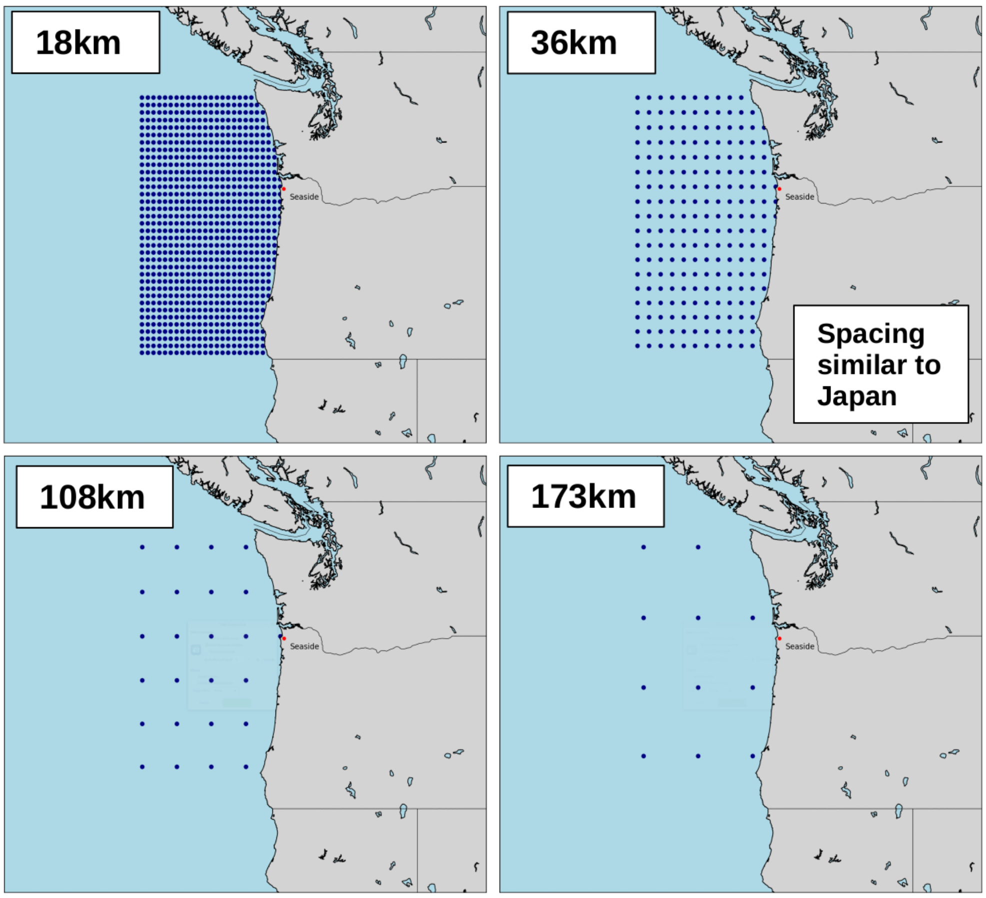


Figure 4.17: 4 of the 18 hypothetical buoy configurations used to create input data for the neural network. Grid spacings range from 18 km to 173 km.

4.4 Machine Learning Techniques

Neural networks first appeared in the 1950s and have seen uses from facial recognition to stock market prediction, now including tsunami inundation forecasting (Mulia et al., 2020; Fauzi & Mizutani, 2020; Makinoshima et al., 2021). Originally inspired by biological neurons in the human brain, artificial neurons were created to mimic the learning process, hence the name artificial intelligence. Although biological and artificial neurons operate in different ways, they both share fundamental characteristics that allow them to learn patterns from a given set of input information.

Artificial neurons mimic the dendrites, cell bodies, and axons found in biological neurons. The dendrites are responsible for receiving electric signals, the cell body then processes the signal, and afterwords this signal is sent to other neurons via the long appendage called the axon. As more neurons are connected together, they form a network where some connections become stronger or weaker depending on the task. What artificial neurons try to do is mimic this network through the structure shown in Figure 4.18.

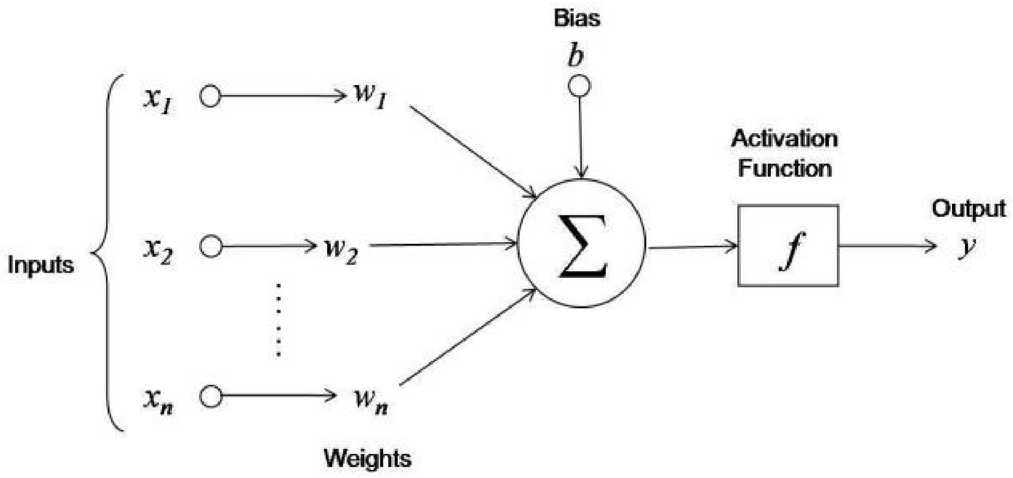


Figure 4.18: One neuron in an artificial neural network. Inputs $(x_1, x_2, ..., x_n)$ are multiplied by their own respective weights $(w_1, w_2, ..., w_n)$ and summed. A scalar bias b is added and the output is fed into a non-linear activation function to produce output y.

First, the neuron receives inputs $[x_1, x_2, x_3, ..., x_n]$ from neurons in the previous layer and multiplies each one by its respective weight $[w_1, w_2, w_3, ..., w_n]$. This is then summed and a bias value b is added. It is represented as

$$z = \boldsymbol{x} \cdot \boldsymbol{w} + b \tag{4.6}$$

where $\boldsymbol{w}$ and bias b are parameters adjusted in the training process.

As shown in the Figure 4.18, the summed weighted input z is then put through an activation function which is responsible for introducing non-linearity to the network. The function itself $f(z)$ can take many forms such as a sigmoid or tanh but for the purposes of this study a Rectified Linear Unit (ReLU) activation function was used (Nair & Hinton, 2010). It is represented by Equation 4.7 and Figure 4.19.

$$y = f(z) = \max(0, z) \tag{4.7}$$

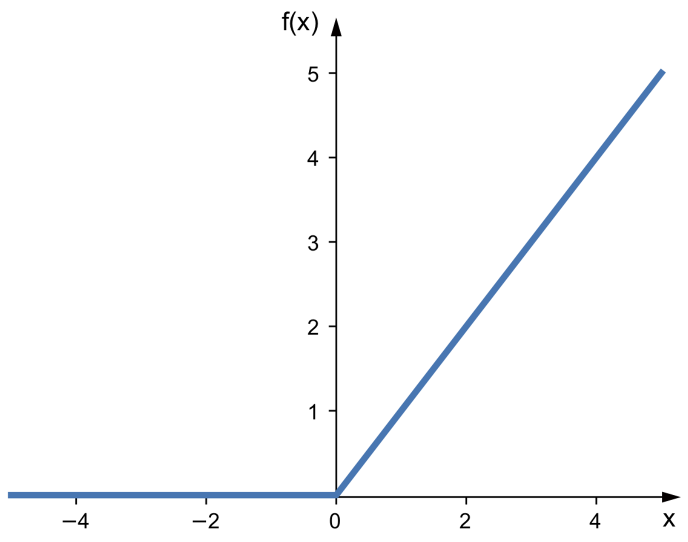


Figure 4.19: The ReLU activation function. An input value x is transformed into an output value $f(x)$, where nonlinearity is introduced in order to enable the network to represent more complicated functions.

The most important purpose of an activation function is to introduce non-linearity to the network. If the sum $\boldsymbol{x} \cdot \boldsymbol{w} + b$ (Equation 4.6) was directly taken as the output, this would inherently restrict the network to linear classifier problems. But to achieve the ability to represent more complexity in the network, a nonlinear activation is necessary.

The ReLU activation function was chosen because it preserves many of the properties that make linear models easy to optimize while still adding non-linearity (Nair & Hinton, 2010). This allows the model to learn faster while also enabling it to learn more complex patterns. For multilayer perceptrons and convolutional neural networks, this is typically the default activation function.

For the simplest case, the artificial neurons, or nodes, are typically arranged in columns, called layers. Every node in a given layer is connected to every node in the layer directly before it, and the outputs of every node in that layer is connected to every node in the layer after it. In this fully connected structure, information is passed from one side of the network to the other, where in the process it is transformed to give the desired output. Depending on the complexity of the dataset, the number of nodes in a column and the number of columns can be adjusted, where more layers and nodes allows for more complexity. As for the output layer, its type depends on the problem statement. Most machine learning problems can be divided into four categories: classification, regression, clustering, and dimensionality reduction. Clustering and dimensionality reduction are mainly used to organize and reduce the size of datasets, and classification type problems are meant to take an input and sort it into a predefined categories. For the problem defined in this study, a regression type output is used as it needs to predict a series of floating point inundation heights. The output layer here is therefore a series of nodes with each representing one pixel of the predicted inundation map.

Once the network is set up and the dataset is organized, the learning process can begin. This starts by splitting the dataset into three sets: the training set, the test set, and the (optional) validation set. The training set is used in the training process, where the weights connecting each node are learned to give the most optimal output. Then, the test set is put through the network to assess the accuracy of a set which the network has never seen. If the network was trained properly on the training set, the accuracy of the test set should be similar. If needed, the optional validation set is used to monitor the accuracy during the training process to assess how well the network is generalizing. Typically, the training set will contain the large majority of the dataset, while the remaining percentage will be split up between the test and validation sets. Out of the 3000 precomputed tsunami scenarios used in this study, 2600 were given to the training set, 300 were given to the test set, and the remaining 100 were used as the validation set.

For regression type machine learning problems such as the one presented here, a metric to assess the accuracy of a prediction is required. During the training process, different weights are assigned to each node and then the output array $y_1, y_2, ..., y_N$ is compared to the expected $\hat{y_1}, \hat{y_1}, ..., \hat{y_N}$. To assess the accuracy of this prediction, the mean squared error is used (MSE). This is also referred to as the cost C.

$$C = \text{MSE} = \frac{1}{N}\sum_{i=1} N(y_i - \hat{y_i})^2 \tag{4.8}$$

The cost function guides the machine learning algorithm to make adjustments to the weights in order to minimize the cost. The most common method of minimizing the cost function is known as stochastic gradient descent. It is an optimization algorithm that calculates the gradient of the cost function in search of a set of parameters which returns the minimum value of that cost function. The method chosen for calculating the gradient of this cost function is known as backpropagation.

The first step to minimizing the cost function is to calculate the gradients using backpropagation. This tells us how the cost function changes when a single weight w_i is changed. It is simply the partial derivative of the cost C with respect to w_i, $\frac{\partial C}{\partial w_i}$. Since it cannot be evaluated directly, it must be first split up using the chain rule.

$$\frac{\partial C}{\partial w_i} = \frac{\partial C}{\partial y_i}\frac{\partial y_i}{\partial z}\frac{\partial z}{\partial w_i} \tag{4.9}$$

For the simplest case we consider a neuron in the output layer. There, w_i represents a weight connected to the neuron, y_i represents the final predicted value, and z represents the summed input fed into that neuron.

The first partial derivative in Equation 4.9 represents how much the cost changes when the output of that neuron changes. It is as follows.

$$\frac{\partial C}{\partial y_i} = \frac{\partial}{\partial y_i}\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{4.10}$$

$$= \frac{2}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i) \tag{4.11}$$

The second partial derivative represents how much the output of the neuron changes when the input changes. The ReLU activation function is used in this calculation as it is the one used in this study.

$$\begin{aligned}
\frac{\partial y_i}{\partial z} &= \frac{\partial}{\partial z} f(z) && (4.12)\\
&= \frac{\partial}{\partial z} \max(0, z) && (4.13)\\
&= \begin{cases} 0 & \text{if } z < 0 \\ 1 & \text{if } z \geq 0 \end{cases} && (4.14)
\end{aligned}$$

The third partial derivative represents how much the sum z changes when a specific weight w_i is changed.

$$\begin{aligned}
\frac{\partial z}{\partial w_i} &= \frac{\partial}{\partial w_i} \sum_{i=1}^{N} w_i x_i + b && (4.15)\\
&= x_i && (4.16)
\end{aligned}$$

Combining these back into Equation 4.9 we obtain

$$\frac{\partial C}{\partial w_i} = \begin{cases} 0 & \text{if } z < 0 \\ \frac{2x_i}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i) & \text{if } z \geq 0 \end{cases} \tag{4.17}$$

For the calculation of the gradient for the bias b in Equation 4.6, $\frac{\partial z}{\partial w_i}$ would be replaced with $\frac{\partial z}{\partial b}$. Since b is a scalar, the derivative of this quantity is always 1.

However, these results represent the gradients for nodes only in the last layer of the network. If we were to calculate gradients for nodes one layer deeper, the calculation would not be as simple. This is because although the partial derivatives $\frac{\partial y_i}{\partial z}$ and $\frac{\partial z}{\partial w_i}$ would still be straight forward to calculate, $\frac{\partial C}{\partial y_i}$ would not as the cost function is not directly related to the output of that node anymore. Instead, the output of the nodes one layer deeper will now be fully connected to every node in the final layer, and the combination of those outputs control the cost function. In other words, the cost function of a node in a hidden layer does not just depend on its own output, but instead it depends on all of the subsequent nodes in the network.

To resolve this, the partial derivative $\frac{\partial C}{\partial y_i}$ can be broken up according to the contributions from every node in the layer ahead of it (in this case, the last layer).

$$\frac{\partial C}{\partial y_i} = \frac{\partial C_1}{\partial y_i} + \frac{\partial C_2}{\partial y_i} + ... + \frac{\partial C_N}{\partial y_i} \tag{4.18}$$

where $C_1,\ C_2, ..., C_N$ represents the cost from nodes 1, 2,...,N in the last layer. Each of these contributions can be rewritten by applying the chain rule. For example, the 1^{st} node's contribution in the last layer can be written as

$$\frac{\partial C_1}{\partial y_i} = \frac{\partial C_1}{\partial z_1}\frac{\partial z_1}{\partial y_i} \tag{4.19}$$

where z_1 is the weighted sum given to the 1^{st} node in the last layer.

In this form, $\frac{\partial z_1}{\partial y_i}$ simply becomes w_1, or the weight connecting node i in the second layer to the 1^{st} node in the last layer. In addition, the partial derivative $\frac{\partial C_1}{\partial z_1}$ becomes $\frac{\partial C}{\partial y_i}\frac{\partial y_i}{\partial z}$ which was already determined from the gradient calculations in the last layer.

This is why the calculation of gradients is called backpropagation. The gradient calculations start from the last layer and work backwards, using previous calculations to determine gradients in the previous layers.

Now that the gradients are calculated using backpropagation, the method of gradient descent can be employed. The procedure follows two straightforward equations which contain a controllable hyper-parameter called the learning rate, denoted by α.

$$w_i^* = w_i - (\alpha \cdot \frac{\partial C}{\partial w_i}) \tag{4.20}$$

$$b^* = b - (\alpha \cdot \frac{\partial C}{\partial b}) \tag{4.21}$$

The quantities w_i^* and b^* represent the updated quantities and are projected to return a network which lowers the cost function.

This process is repeated until ideally the a minimum of the cost function is found. Although often times, the learning rate is either too high or too low to locate the minimum and instead settles on a local minimum. The learning rate therefore needs to be carefully adjusted to achieve the best performance of the network.

The methods described above form the foundation of modern machine learning algorithms and allow complicated patterns such as tsunami inundation to be captured. The two types of neural networks used in this study, multilayer perceptrons for the ground deformation data and convolutional neural networks for the buoy data, will be discussed in more detail in the following sections.

4.4.1 Multilayer Perceptron

For the purpose of converting the ground deformation data into an inundation forecast, a multilayer perceptron (MLP) was chosen as the neural network model and was implemented in PyTorch (Paszke et al., 2017). MLPs are models that work as universal approximators, meaning they can approximate any continuous function (Hornik, 1991). This model is almost identical to what is described above in the previous section, the only difference being that the aforementioned neurons are called perceptrons (Rosenblatt, 1957). In an MLP, the neurons, or perceptrons, are arranged in layers, where each layer is fully connected to the ones before and after it. The general structure consists of an input layer, one or more hidden layers, and an output layer.

Here, the input layer consists of a list of values corresponding to vertical ground displacement at selected points. The output layer consists of a list of maximum flow depths corresponding to each pixel in the inundation map. In between the input and output layers are two hidden layers of constant size.

There are several things to consider when choosing the number of layers. J. Heaton (2008) states there is no theoretical reason to use any more than two hidden layers in a neural network since these two layers can represent functions of any shape. However, using more layers but less nodes per layer could prove more efficient as they could represent the same function with fewer learnable parameters. On the other hand, too many layers may overfit the data and essentially “memorize” the training dataset. Ultimately, it was found through testing that two layers was both sufficient and efficient to represent the function from ground data to inundation heights.

Next, the number of nodes in each hidden layer was chosen. This parameter can heavily affect the overall performance and as a result there exists a number of guidelines to determine how many nodes to use. According to J. Heaton (2008), some of these rules include: the number of hidden neurons should be in between the size of the input and output layers, the number of hidden neurons should be two thirds the size of the input layer added to the size of the output layer, etc. However, these are merely rules-of-thumb and the number of neurons must ultimately be decided through trial and error. The tests here include varying the length of the two hidden layers from 50 nodes to 400 nodes. This range of nodes was determined through preliminary testing where it was found that lengths outside this range would not significantly alter the results. The concern with a layer size too small is that it leads to underfitting, meaning the complexities of the dataset are not fully captured by the network. With a layer size too large, it leads to overfitting, meaning the network is so large that it memorizes the dataset. The goal of varying the layer size is to find a network size which captures the complexities of the dataset while also generalizing the problem.

Figure 4.20 shows the MLP for ground deformation data from existing GNSS stations along the coast.

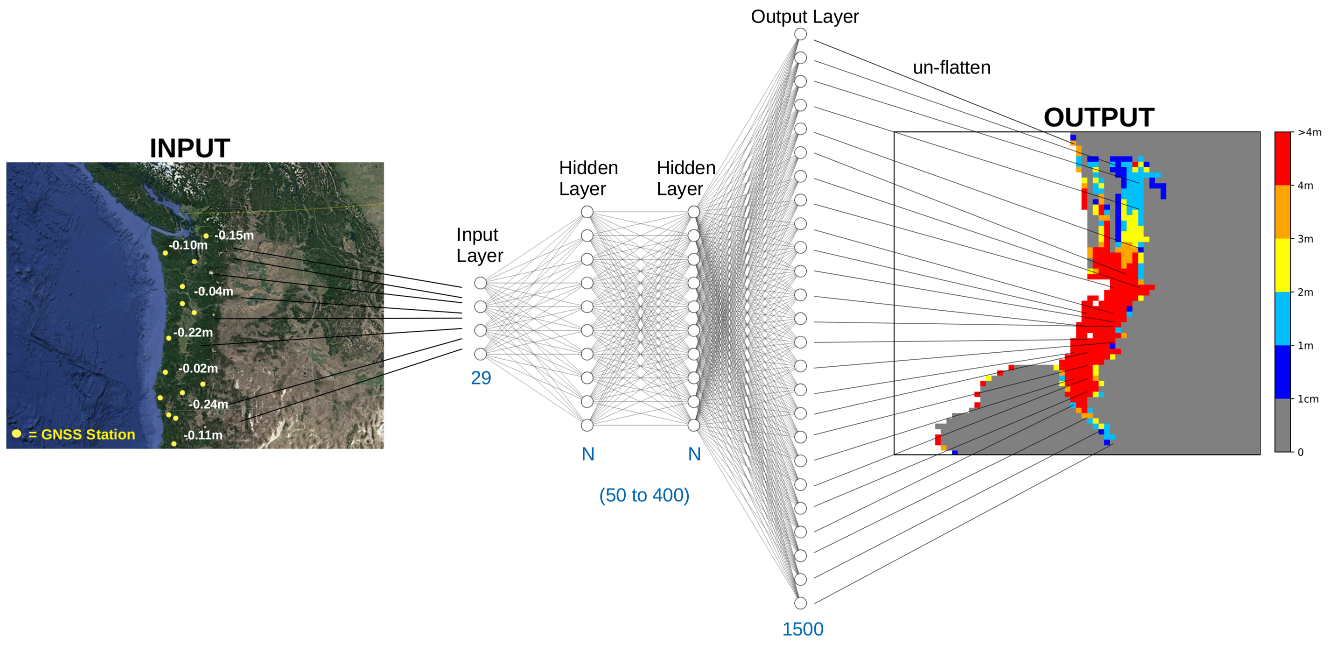


Figure 4.20: MLP design consists of an input array of 29 existing ground deformation measurements followed by two hidden layers which produce an output array of length 1500. Once un-flattened, the forecasted inundation map can be seen. The length of the hidden layers are adjustable and were varied to explore the effect on accuracy.

Figure 4.21 shows both the same hidden layers and output layer as the MLP in Figure 4.20, with the exception of a different input layer. To test the capabilities and limitations of inundation prediction using ground deformation data, multiple hypothetical grids of observation points were used as input. Here, points were not limited to onshore locations, but rather sampled across the entire simulated earthquake region. The total number of input points therefore ranges from 500 to 4500.

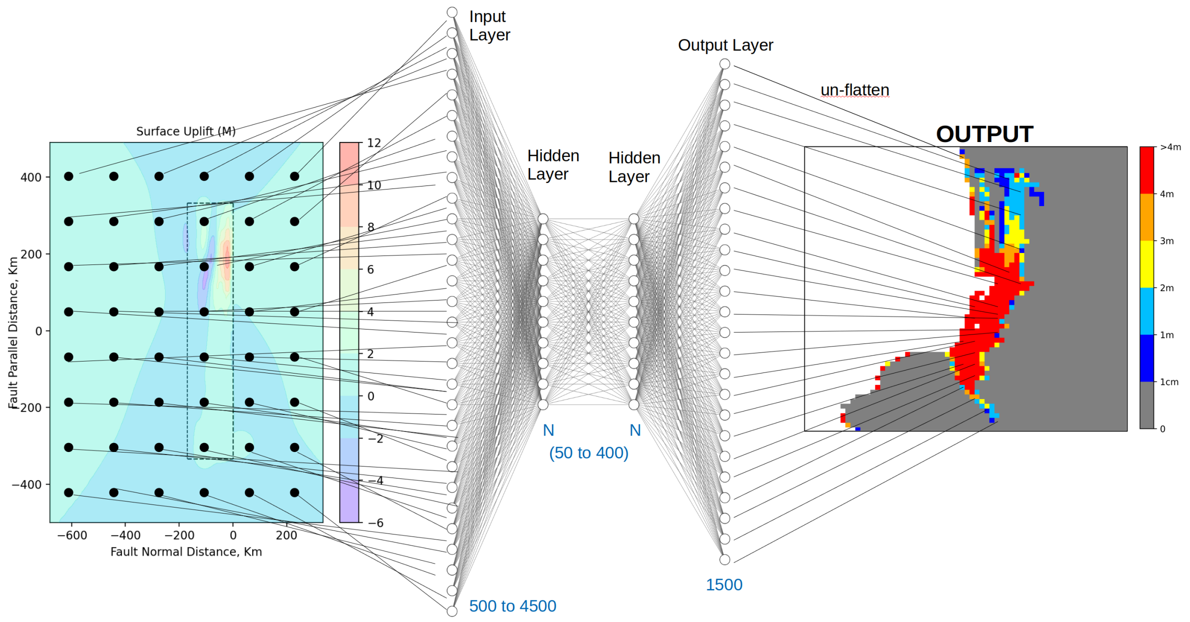


Figure 4.21: MLP design consisting of several input lengths where vertical deformation points were sampled over the entire rupture region. The amount of input observation points varies from 500 to 4500 increasing in 500 point increments to explore the capabilities of such a method. Similar to Figure 4.20, the hidden layer length was varied to explore effects on accuracy.

The tests mentioned above were carried out in two separate experiments, one using a high learning rate and one using a low learning rate. The number of epochs (learning iterations) were adjusted according to the two learning rates such that the lower learning rate is given more epochs, and the higher learning rate is given less. The learning rates were chosen to be 0.05 and 0.01 with 100 and 500 epochs respectively.

The number of input points, nodes, learning rate, and epoch number were all systematically adjusted to explore the capabilities of an MLP to give accurate inundation predictions based on ground deformation data. The results of these tests will be discussed in further sections.

4.4.2 Convolutional Neural Network

In order to convert the buoy data into an inundation forecast, a convolutional neural network (CNN) was utilized and was implemented in PyTorch (Paszke et al., 2017). CNNs specialize in pattern recognition and image classification (Lo et al., 1995; Lawrence et al., 1997; Ciregan et al., 2012; Garcia & Delakis, 2004; Sermanet et al., 2013; Russakovsky et al., 2015). It has seen uses in facial recognition, natural language processing, X-ray image analysis, object detection, and many other areas. A CNN was chosen mainly for its potential to detect patterns in the time series arrays containing wave height data (buoy data) used in this study. It offers a highly efficient alternative to an MLP, which assigns weights to every bit of information, regardless of if its importance. A CNN utilizes a moving filter which acts to prioritize the important features of a dataset, minimizing the amount of learnable parameters needed.

The core building block in a CNN is the convolutional layer which consists of a small filter, or kernel, which scans the input to produce a unique output. In a 2D convolutional layer, the typical kernel size is 3x3 where each element is a learnable parameter and is slid over the input and convolved to produce a new output. For an input with multiple channels such as the buoy data (one buoy's time series array represents one channel), the filter will extend through the full depth of channels. For example, if a 1D kernel of length three were used on an input source consisting of 15 buoys, there would be $3 \cdot 15 = 45$ learnable parameters in that layer. The mechanics of the convolutional layer is best understood in terms of a 2D filter which is shown below in Figure 4.22.

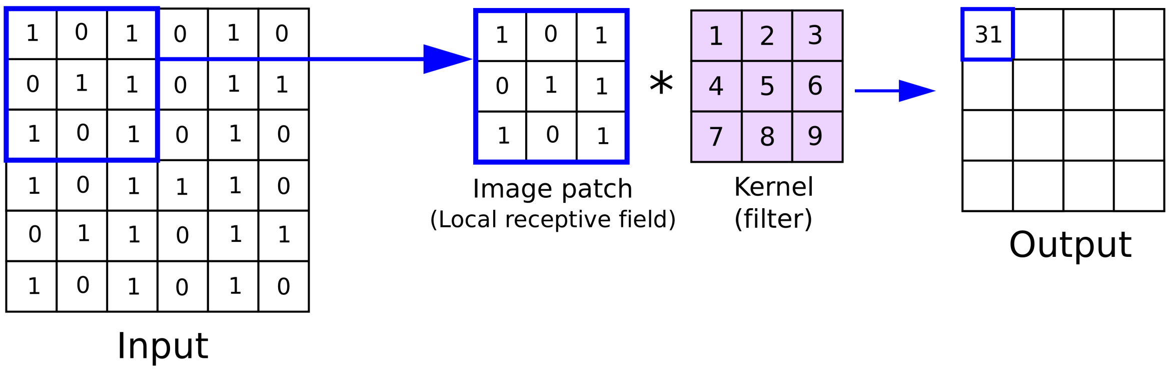


Figure 4.22: The convolutional layer procedure. A kernel is superimposed onto the input grid in which every overlapping value is multiplied and then summed. This represents one value of the output array. The kernel is slid over the entire input array until all values of the output array are filled.

In Figure 4.22, an image patch is selected and multiplied by the kernel. In this case the kernel is a 3x3 matrix where each value in the matrix is a learnable parameter. The product of each element is then summed to give the output of 31 seen in the upper left hand corner of the output array. The kernel is then slid over the entire 2D grid of input values to produce a new 2D grid of output values.

In Figure 4.22, the kernel is shifted one element at a time. The amount of elements the kernel is shifted is known as its stride. A stride of two would mean the kernel is shifted two elements before being multiplied with the image patch. A larger stride effectively decreases the size of the output matrix and can be used as a way of downsampling the data as was done for several layers in the CNN used in this study.

The kernel size can also be adjusted depending on the type of problem and can affect the dimensions of the output array. For example, if a larger kernel of size 5x5 was used in Figure 4.22, the output matrix would be of size 2x2. This may be a problem as the edges of the matrix are not being represented by the filter properly, and are effectively being cut off. A solution to this is to introduce padding around the border of the input matrix allowing the filter to go beyond the edges. The padding around the edges is usually set to 0 to minimally affect the results. Padding was used in the construction of the CNN used in this study to remedy the unwanted changes in output size.

In Figure 4.22, the 2D input matrix uses one kernel to produce one 2D output matrix. In other words, the one-channel input uses one kernel to produce a one-channel output. For a given input, multiple kernels can be used to produce a multi-channel output. The idea behind this is that multiple kernels are able to pick out multiple, distinct features in the data. This concept is utilized in the CNN constructed for this study.

Although the convolutional layers in CNNs are structurally different than the fully connected layers present in MLPs, they both utilize backpropagation and gradient descent in fundamentally the same way. The only difference being the convolutional layers have considerably less partial derivatives to calculate.

The general CNN architecture consists of layers constructed in the following order: an input layer, one or more convolutional layers with pooling layers mixed in, two or more fully connected layers, an output layer. The CNN architecture developed for this study follows this general structure and can be seen below in Figure 4.23.

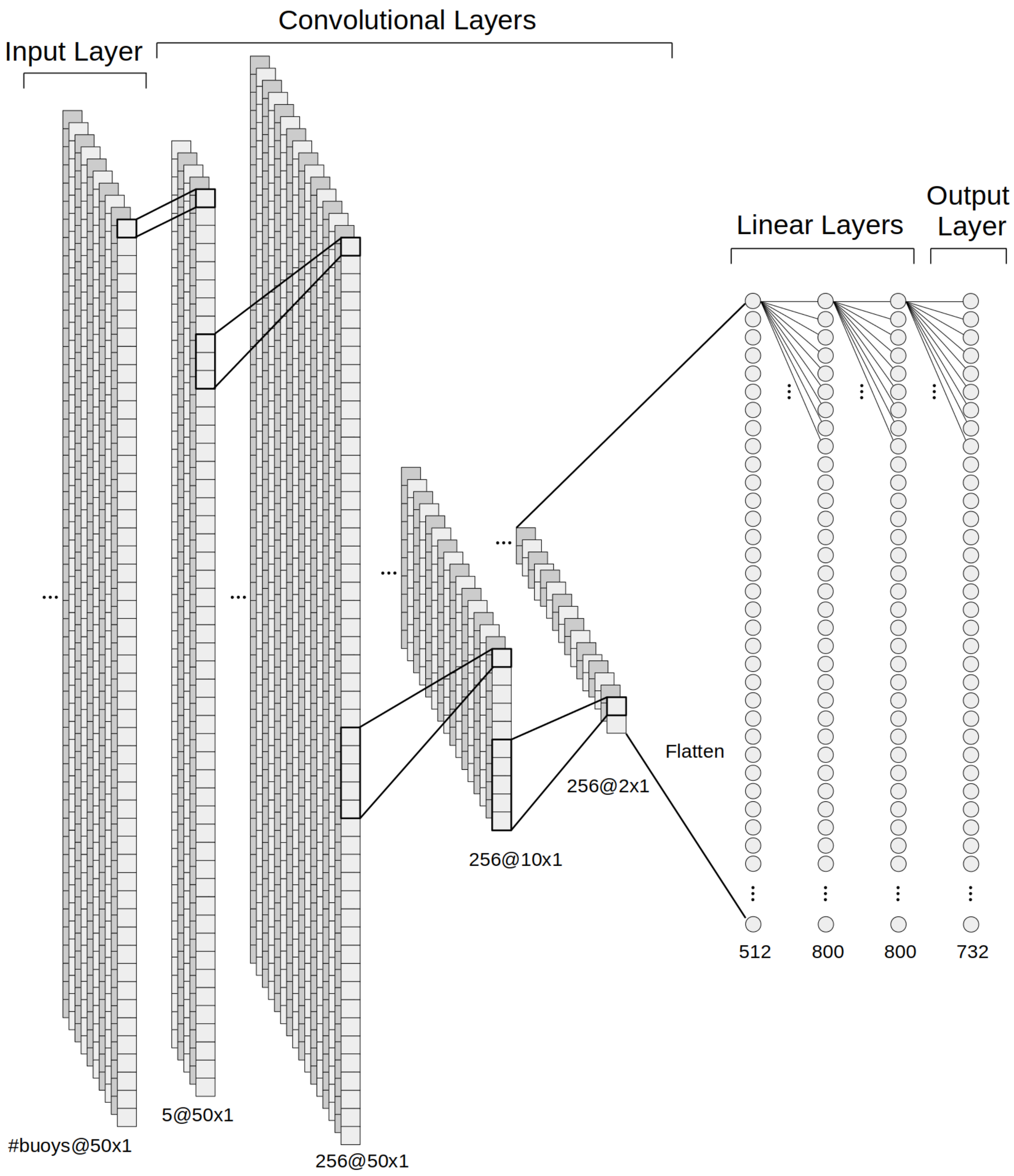


Figure 4.23: The CNN architecture. Each column in the input layer consists of the time series from one buoy, where the depth (or number of channels) equals the number of buoys. Data is passed through the subsequent convolutional layers where it is reduced to a flattened array of length 512. This is then passed on to two fully connected layers where a flattened inundation map is produced.

Layer	**Kernel Size**	**Stride**	**Padding**	**Input Channels**	**Output Channels**
Conv	1	1	0	# Buoys	5
Conv	3	1	1	5	256
Conv	5	5	0	256	256
Conv	5	5	0	256	256
Linear	None	None	None	512	800
Linear	None	None	None	800	800
Output	None	None	None	800	732

Table 4.2: The parameters of each layer for the CNN presented in Figure 4.23. In between each layer is a ReLU function.

The CNN architecture in Figure 4.23 was inspired by the neural networks found in Makinoshima et al. (2021) and Toledo-Marín et al. (2021), but was ultimately determined by trial and error. Since each problem is different, there exists no network which can fit all applications, even if the problem domain is similar.

The CNN in Figure 4.23 consists of four convolutional layers and two fully connected layers. The first convolutional layer has a kernel size of 1x1 and is meant to act as a filter across channels (Lin et al., 2013). Since the kernel extends down through all channels, the weights assigned to the 1x1 kernel are equivalent to a weight assigned to the channel itself. If the optimization algorithm decides one channel (or one buoy) is particularly important, it will give that channel a high weight. In essence, the 1x1 convolutional layer used here takes the number of input channels (or input buoys) and distills them to five channels. Since the input number of buoys was varied from 10 to 800, the addition of this type of convolutional pooling layer was critical to ensure the following layers will receive a consistent amount of information as input. This removes the neural network structure as a confounding variable in the comparison of accuracy between differently sized inputs.

The following layer uses a standard kernel size of three with a padding value of one to preserve the length of each channel. This layer acts as the main pattern finding layer and outputs 256 channels of length 50. The following two layers are convolutional pooling layers and act to downsample the data. Usually in neural networks, a maximum pooling operation is used which returns the maximum of every N elements, effectively dividing the size by N. Makinoshima et al. (2021) achieves the same result in a convolutional pooling layer where the kernel size and stride are set to the same number N to ensure that the kernel will return a value every N elements. In the CNN used here, the kernel and stride were both set to five to downsample the length from 50 to 10 in the first layer, then 10 to 2 in the second layer. The convolutional pooling layer approach to downsampling was chosen over the traditional maximum pooling operation solely due to an empirical improvement in performance. The 256 channels of length two are then flattened into a 1D array of 512 elements and then fed into the two linear layers of length 800, where it is finally transformed into a 732 pixel inundation map by the output layer. To prevent overfitting, dropout with a probability of 0.001 was implemented directly before the output layer. Dropout refers to the process of temporarily ignoring, or "dropping out" different nodes in a given layer during each epoch of the training process. The network is therefore training on a different randomly generated thinned version of the network at every epoch. This has the effect of combining many different neural network architectures together to produce an averaged output, allowing the network to generalize well as it approximates an ensemble approach (Srivastava et al., 2014).

Regarding the hyperparameters of the CNN, a learning rate of 0.0005 was chosen and trained for a total of 800 epochs. At the end of the 800 epochs, the weights from the epoch with the best validation accuracy was chosen. The cost function was chosen to be MSE as it performed better compared to the L1 cost function (least absolute error) and the Adam Optimizer Algorithm (Kingma & Ba, 2014) was used for gradient descent.

4.5 Results Using Ground Deformation as Input

Here we discuss the inundation forecasting results from the ground deformation data used as input for the MLP. Tests were carried out by varying the number of GNSS observation points, the MLP layer size, and the learning rate in order to determine the combination with the most accuracy given the least amount of observation points needed. Figure 4.24 shows the raw results for six randomly selected events from the test set using the existing 29 GNSS station observation points as input.

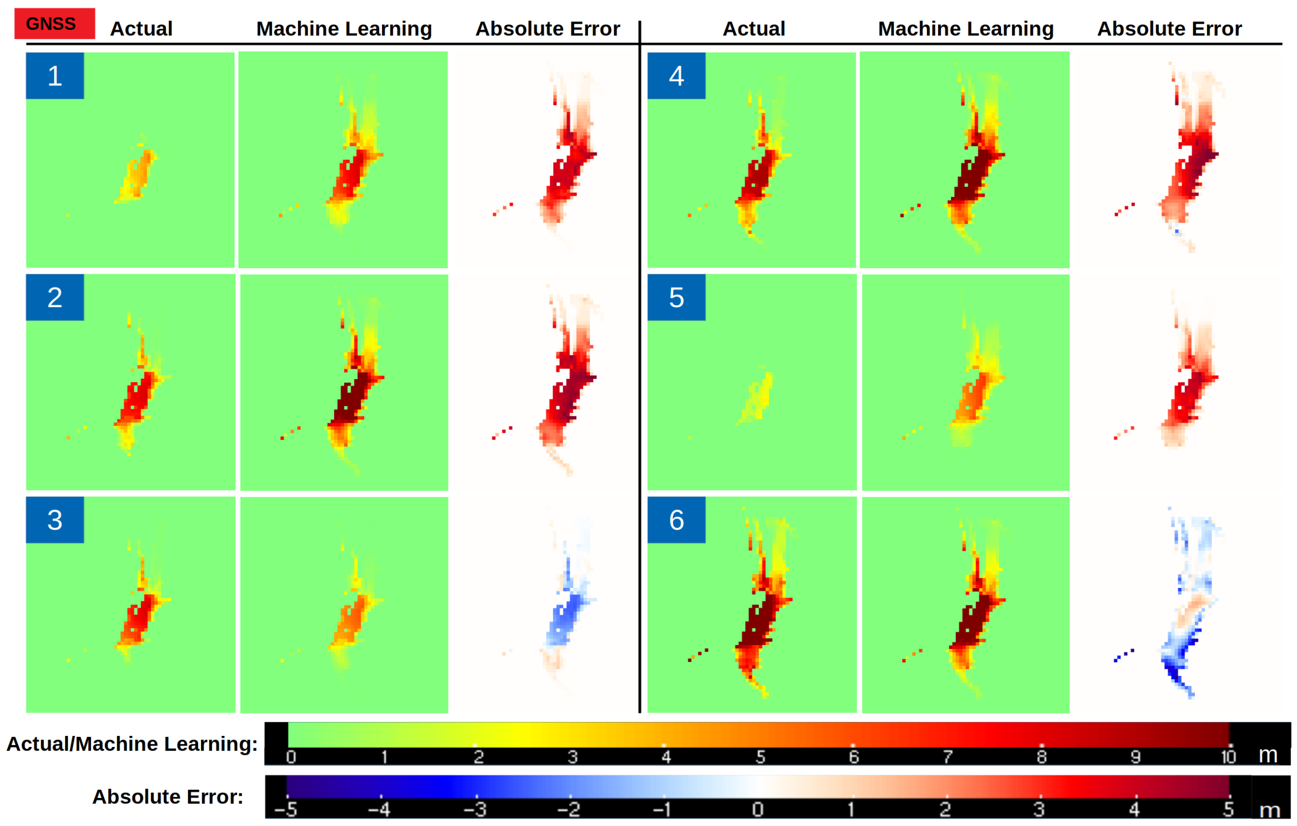


Figure 4.24: Results for six randomly selected events in the testing dataset using 29 ground deformation observation locations as input. Actual inundation maps refer to the forward modeling result for that event.

It can seen in Figure 4.24 that for nearly all of the six randomly chosen events, the absolute error exceeds ±5m. This particular test uses a layer size of 400 and a learning rate of 0.01. However, for all of the tests using existing GNSS locations, performance is similarly poor regardless of the MLP layer size or learning rate.

Figure 4.25 shows the results for the configuration which returned the highest accuracy with the least number of observation points needed. The parameters used for this test are as follows: 3500 input points, 400 nodes in each layer, and a learning rate of 0.05.

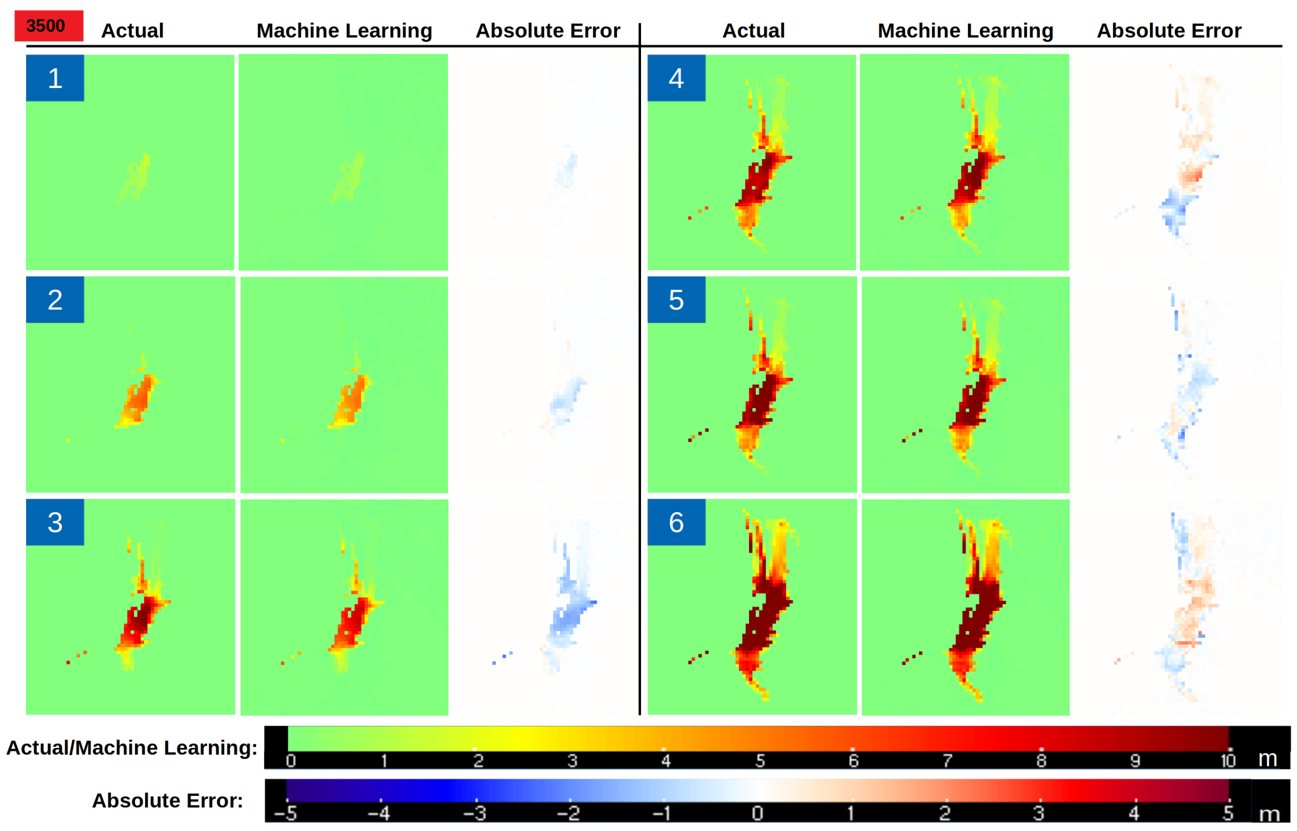


Figure 4.25: Results for six randomly selected events in the testing dataset using 3500 ground deformation observation locations as input. Predicted inundation maps generally agree with expected inundation as the majority of the error falls between -1 m and +1 m.

From Figure 4.25 it is clear that using a significantly greater amount of observation points allows the MLP to produce more accurate inundation predictions compared to the results using the existing 29 GNSS stations. The maximum error is confined to $\sim\pm 2$m in these six examples.

Results for the average error, the average difference in coverage, and a metric for overfitting are discussed here. The average error is first calculated per event, then the error of each event is averaged to get a total error for the entire test set. However, because the inundation maps contain many predicted and expected values of zero, considering these points will result in a very low error calculation. Therefore, the inundation heights which are less than 15cm in either the expected or predicted inundation maps are ignored. The cutoff of 15cm was chosen because the NOAA National Weather Service reports that 15cm of water is sufficient to knock over a standing person or to stall a car. This cutoff was also used in a study by Priest et al. (2016).

The equation used to calculate average event error is given as

$$E_j = \frac{1}{N_j}\sum_{i=0}^{N_j} |h_i - \hat{h}_i| \qquad \text{if } \hat{h}_i > 0.15 \text{ m or } h_i > 0.15 \text{ m} \tag{4.22}$$

where E_j is the average error for the j^{th} event in the testing set, $\hat{h}_i$ is the expected inundation height of the i^{th} pixel, h_i is the predicted height of the i^{th} pixel, and N_j is the total number of pixels in event j which satisfy $\hat{h}_i > 0.15$ m or $h_i > 0.15$ m.

To obtain the average error for the entire testing set, the quantities in Equation 4.22 are calculated for every event in the testing set and then averaged. This operation is defined as

$$E_{avg} = \frac{1}{n}\sum_{j=0}^{n} E_j \tag{4.23}$$

where n is the number of events in the test set (300 for this study). The average error is shown in Figure 4.26 for all tested combinations of the number of input points, MLP layer size, and learning rate.

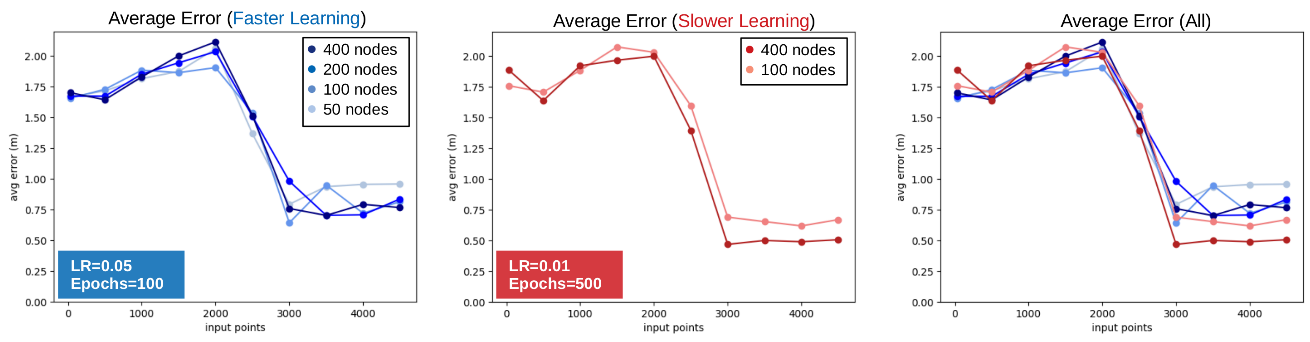


Figure 4.26: Average error results showing a sharp decrease in error after 2000 input points. Left: A faster learning rate tested with four hidden layer lengths. Center: A slower learning rate tested with two hidden layer lengths. Right: Results from the left and center plots overlaid.

In Figure 4.26, general pattern in accuracy can be seen across the number of input data points used. Using a smaller number of data points yields an average error of ∼2m which steadily increases until 2000 points are used. After which, the error steeply declines until 3000 data points are used where it flattens to an error of ∼0.75 m. The sharp decrease in error can be attributed to the neural network requiring a minimum amount of information in order to make a more accurate prediction. At this point the network is saturated as any additional information about the earthquake displacement pattern adds no additional accuracy.

Regarding the number of nodes used in each layer of the MLP, less of a pattern can be seen. On average, lower error is achieved with a larger number of nodes.

The learning rate also has an effect on the average error. It can be seen in the plot on the right in Figure 4.26 that for the range of 3000-4500 input points, a lower learning rate performs better.

Figure 4.27 shows a similar pattern for the average difference in coverage. The coverage for a given event is defined as the number of pixels with an inundation height greater than 15cm. The absolute value of the difference in coverage between the predicted and expected result is then calculated for each event and then averaged, similar to how the average error is defined in Equation 4.23.

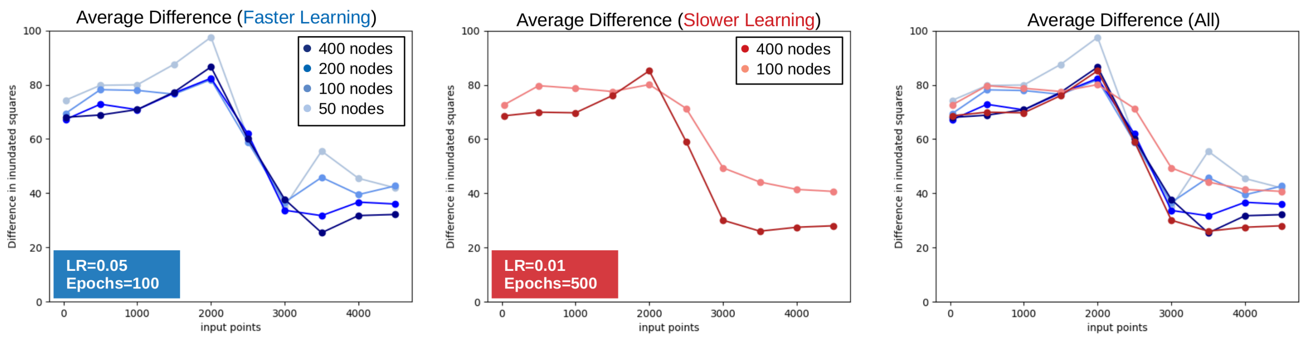


Figure 4.27: Average difference in pixels between predicted and expected coverage. Left: A faster learning rate tested with four hidden layer lengths. Center: A slower learning rate tested with two hidden layer lengths. Right: Results from the left and center plots overlaid.

The same general pattern across the number of input points as in the average error plots. The average difference in coverage starts at ~80 pixels for the low range of input points and sharply decreases to ~40 pixels at 2000 input points. It then plateaus for higher numbers of input points. Regarding the number of nodes and learning rate, the comparisons are similar to that of the average error plots.

The following three plots show an analysis of overfitting using a metric defined by $\frac{E_{\text{test}}}{E_{\text{train}}}$. A value of one would indicate no overfitting at all. However, a value of two or three would indicate strong overfitting as the error of the test set would be two or three times higher than the training set. It is an indication that the model is not generalizing well and is therefore undesirable and must be factored in when finding the best model.

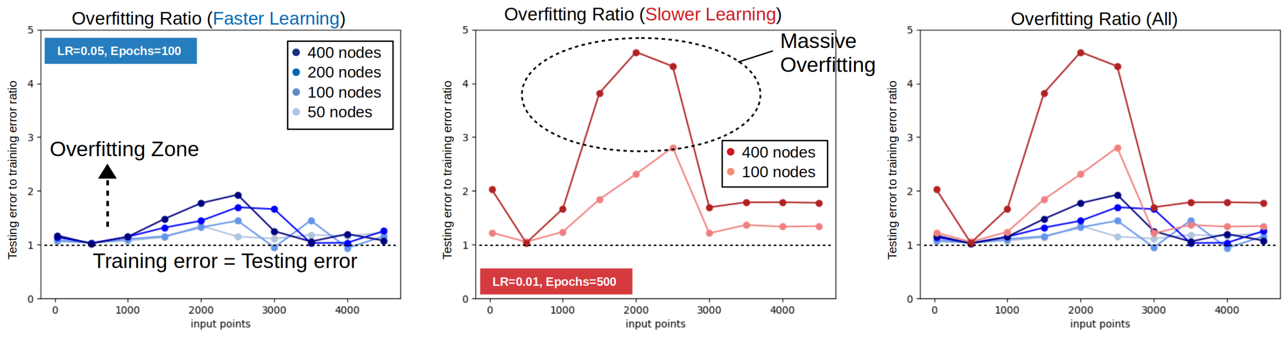


Figure 4.28: Overfitting ratio defined by $\frac{E_{\text{test}}}{E_{\text{train}}}$. Left: A faster learning rate tested with four hidden layer lengths. Center: A slower learning rate tested with two hidden layer lengths. Right: Results from the left and center plots overlaid. A slower learning rate displays a significantly higher overfitting ratio than the higher learning rate.

The left plot in Figure 4.28 shows all tests falling below an overfitting ratio of 2, indicating overfitting is not very strong for a faster learning rate. However, the center plot shows strong overfitting for a slower learning rate. It is also important to compare the range of input points from 3000 to 4500 as these tests have the greatest accuracy. The faster learning rate shows little to no overfitting while the slower learning rate shows appreciable overfitting in this range.

Based on these results, we choose the model using 3500 input points, 400 nodes, and a faster learning rate as it exhibits the greatest accuracy given the least amount of input points and overfitting. Results of this configuration for six randomly selected events can be seen in Figure 4.25. By contrast, Figure 4.24 shows results using the 29 existing GNSS locations and has an average error of $\sim$1.75 (similar to input points in the range of 500-2000). It is clear that using a low number of input points leads to inaccurate inundation forecasts and only until a threshold of 3000 input points do we see forecasts approaching an acceptable amount of accuracy.

Although the methods used here show promise when using 3500 ground deformation observation points, it would be impractical to build and maintain such a large number of stations. It can therefore be concluded that an extremely large amount of GNSS stations are needed in order to achieve accurate inundation predictions for Seaside, OR when using a MLP (Figure 4.21) with a database consisting of 3000 earthquake scenarios near the site of interest. However it is not yet clear whether or not a more sophisticated neural network design or an increase in the number of training events could reduce the number of GNSS stations required to produce an accurate forecast. More testing would need to be done in order to rule out these possibilities and conclude if this method is viable.

4.6 Results Using Buoy Data as Input

Here we discuss the inundation forecasting results from buoy data used as input for the CNN. In addition to using the existing buoys located off the coast of Oregon, several hypothetical arrangements of buoys were also used to test the capabilities of this method. The hypothetical arrangements consist of grids of buoys spanning the simulated region, each with their own buoy-to-buoy separation distance. In total, 18 grids were tested with separation distances ranging from 18 km to 171 km as shown in Figure 4.17.

Figure 4.29 shows three plots containing results for the average absolute error, the average percent error, and the percent difference in coverage. As with the results from the ground deformation data in the previous section, both predicted and expected inundation heights below 15 cm are ignored to maintain a practical measure of error. Otherwise, a measure of the average error will likely return falsely accurate since the measure will be diluted by a large number of matching predicted and expected heights near zero. The method by which the average error per event is calculated is identical to that of the previous section defined by Equation 4.22. The percent error for each event j is defined as

$$\text{Percent E}_j = \frac{100}{N}\sum_{i=0}^{N}\frac{|h_i - \hat{h}_i|}{\hat{h}_i} \qquad \text{if } h_i > 0.15 \text{ m} \tag{4.24}$$

where $\hat{h}_i$ is the expected inundation height of the i^{th} pixel, h_i is the predicted height of the i^{th} pixel, and N_j is the total number of pixels in event j which satisfy $h_i > 0.15$ m.

The percent difference in coverage for each event j is defined as

$$\text{Percent Difference}_j = 100 \cdot \frac{|n - \hat{n}|}{\hat{n}} \tag{4.25}$$

where $\hat{n}$ represents the total number of inundated pixels in the expected inundation map and n represents the total number of inundated pixels in the predicted inundation map for a given event j in the testing set. An inundated pixel is defined as one with a maximum height greater than 0.15 m.

Instead of simply averaging the errors from all 300 events in the testing set, the 10^{th}, 50^{th}, and 90^{th} percentiles were calculated to give a more accurate representation of the error distribution. For reference and comparison purposes, the results using the existing buoys are shown in the form of black horizontal lines.

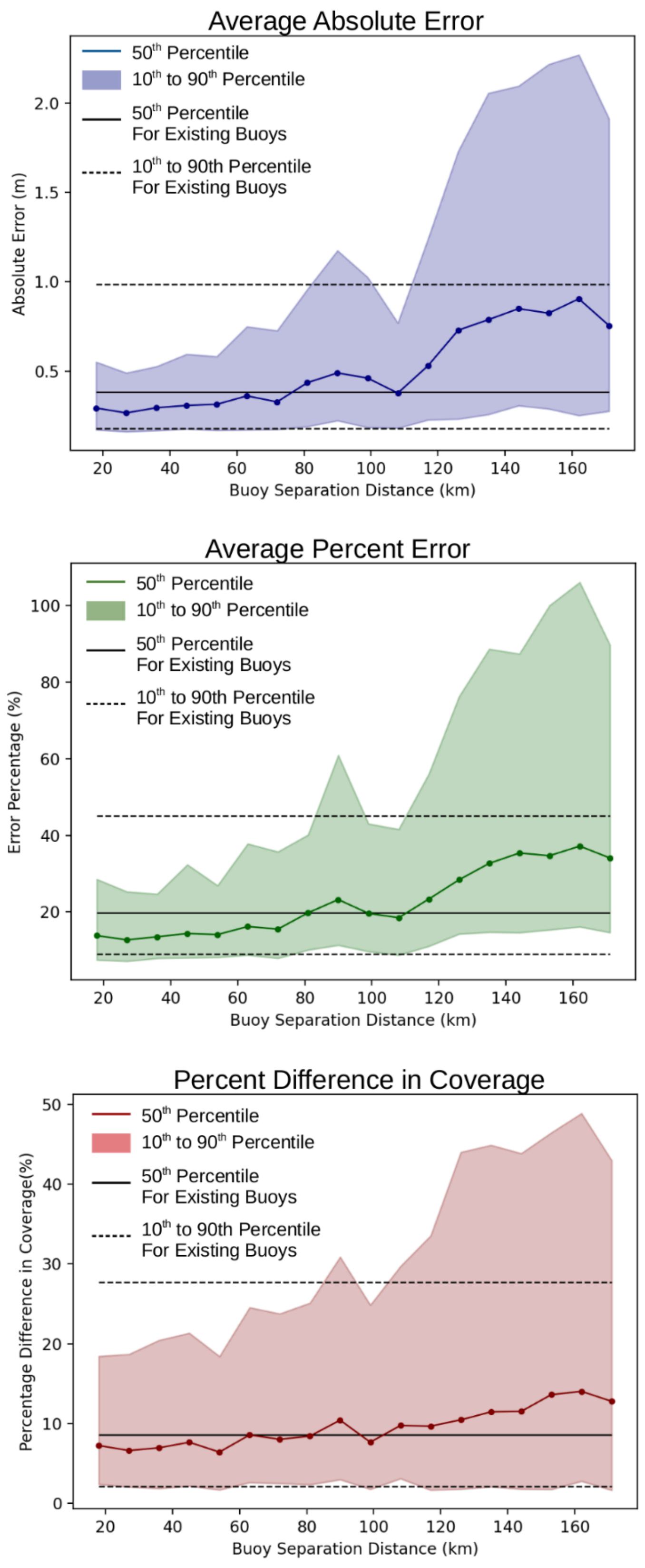


Figure 4.29: Measures of error as a function of buoy separation distance (increasing sparseness). Percentiles for the results from existing buoy locations are indicated by solid and dotted black lines. Results most notably indicate 90th percentiles increase significantly with increasing buoy sparseness.

The average absolute error plot shown in Figure 4.29 shows a clear upwards trend in the 50th percentile. This is expected since as the buoy separation distance increases, the total number of buoys decreases, meaning the CNN has less data from which to make a prediction. A tight grid of buoys (separation distance 20-40 km) produces a median error of ∼0.25 m and the existing buoys produce a slightly higher median error of ∼0.35 m. In fact, the median error of the grids outperform existing buoys until a separation distance of ∼80 km. However, the difference between 0.25 m and 0.35 m error is not very significant. Focus is therefore given to the 90th percentile, which is ∼0.5 for tight grids and ∼1.0 for existing buoys. For larger buoy separation distances, the 90th percentile increases dramatically to over 2m. This means for configurations with low numbers of buoys (existing buoys and 100-180 km separation distances), the chances of an erroneous prediction are significant.

As with the average absolute error plot, the percent error results show a steadily increasing trend in the 50th percentile with a large jump in the 90th percentile at higher separation distances. A tight grid of buoys produces a median percent error of ∼15% while the existing buoys produces ∼20%. However, although the median percent errors are similar, the 90th percentile of the existing buoys is significantly higher at ∼45% compared to the tight grid configurations at 35%. This indicates that a tight grid of buoys will ensure accurate predictions for almost all events in the testing set while the existing buoys have the potential to produce a prediction with nearly 50% error.

The percent difference in coverage is mainly consistent at the 50th percentile with ∼10% difference. The only significant trend can seen in the 90th percentile where it generally increases from 20% to 50%.

An interesting feature of plots in Figure 4.29 can be seen in the form of dips in the error present at separation distances 110 km and 170 km. This is due to some grid configurations with buoys positioned very near the point of interest. These buoys offer critical information the CNN needs in order to make a more accurate inundation prediction, thus decreasing the error for grid configurations with these buoys. This is made clear in Figure 4.30, where the distance of the closest buoy is plotted and aligned with the average absolute error shown in Figure 4.29.

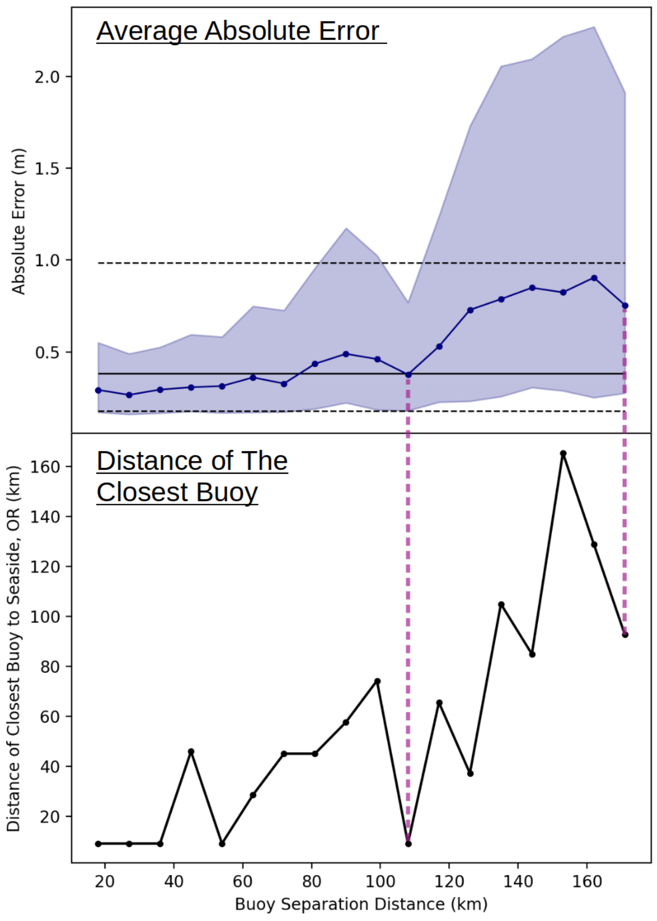


Figure 4.30: Dips in error from Figure 4.29 are caused by the placement of a hypothetical buoy very near to the site of interest. Distance of the closest buoy is defined as the smallest distance between a given buoy to Seaside. The dotted lines show the connection between the distance of the nearest buoy to the dips in error.

In Figure 4.30, the dips in the error are strongly associated with dips in the distance of the closest buoy, indicating the CNN's strong sensitivity to buoys near the site of interest.

Figures 4.31, 4.32, and 4.33 show forecasts for six randomly selected events from the test set for existing buoys, a grid with an 18 km separation distance, and a grid with a 162 km separation distance. This is meant to show the capabilities of the buoys already in operation compared to a tight and loose grid. The tight grid shows what the CNN is capable of given a large amount of information.

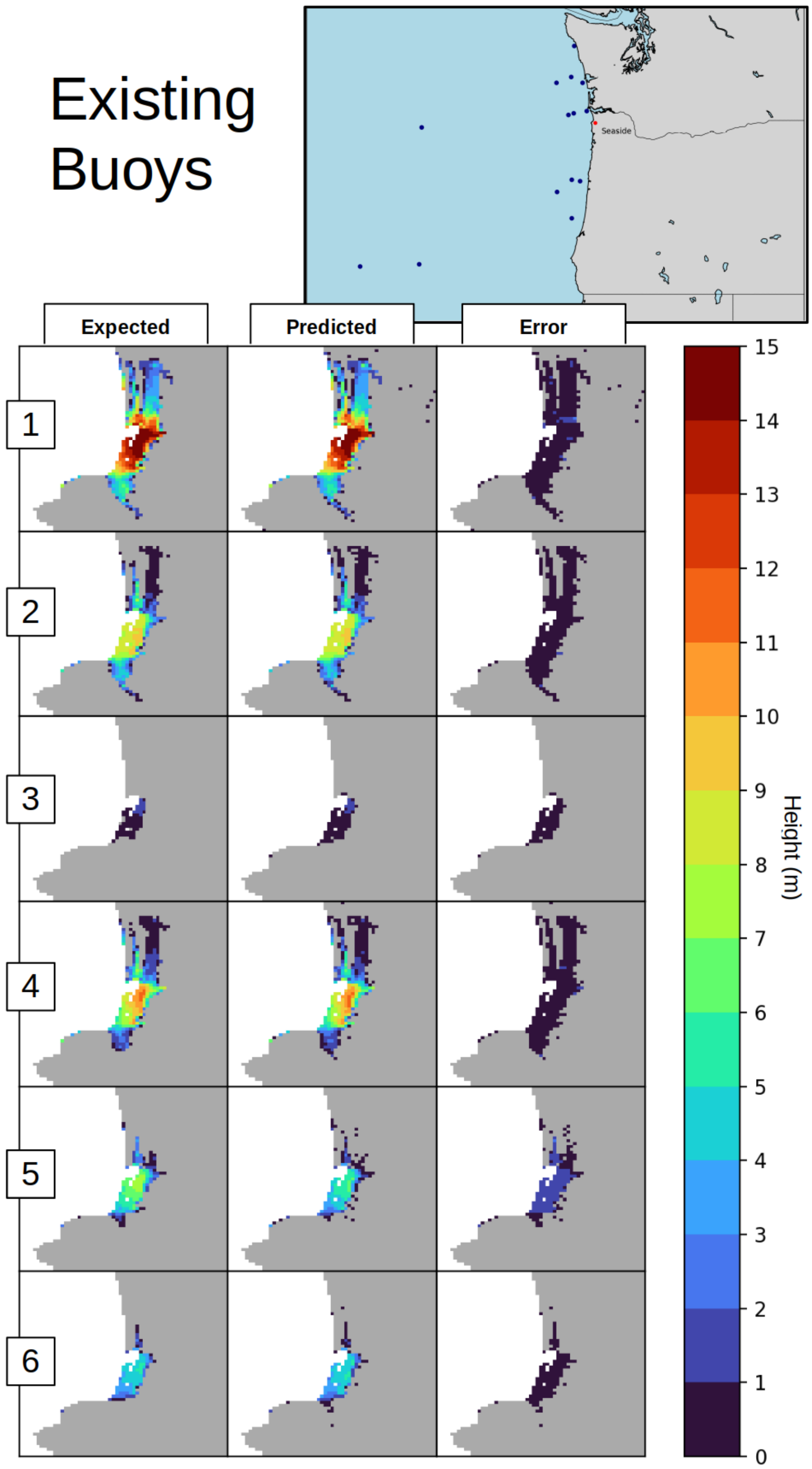


Figure 4.31: Results for six randomly selected events in the testing set using existing buoys.

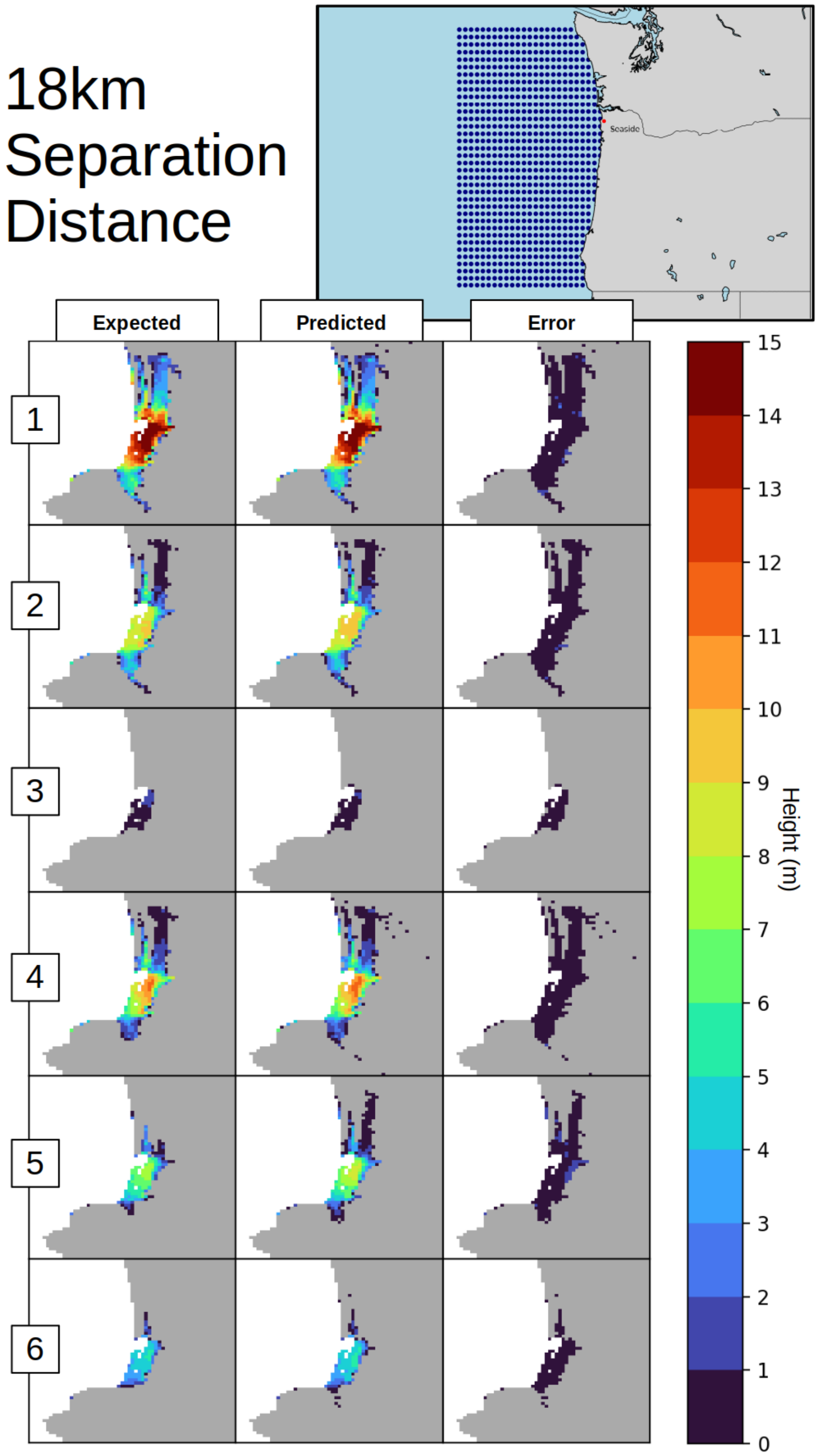


Figure 4.32: Results for six randomly selected events in the testing set using a hypothetical grid of buoys with 18 km separation distance.

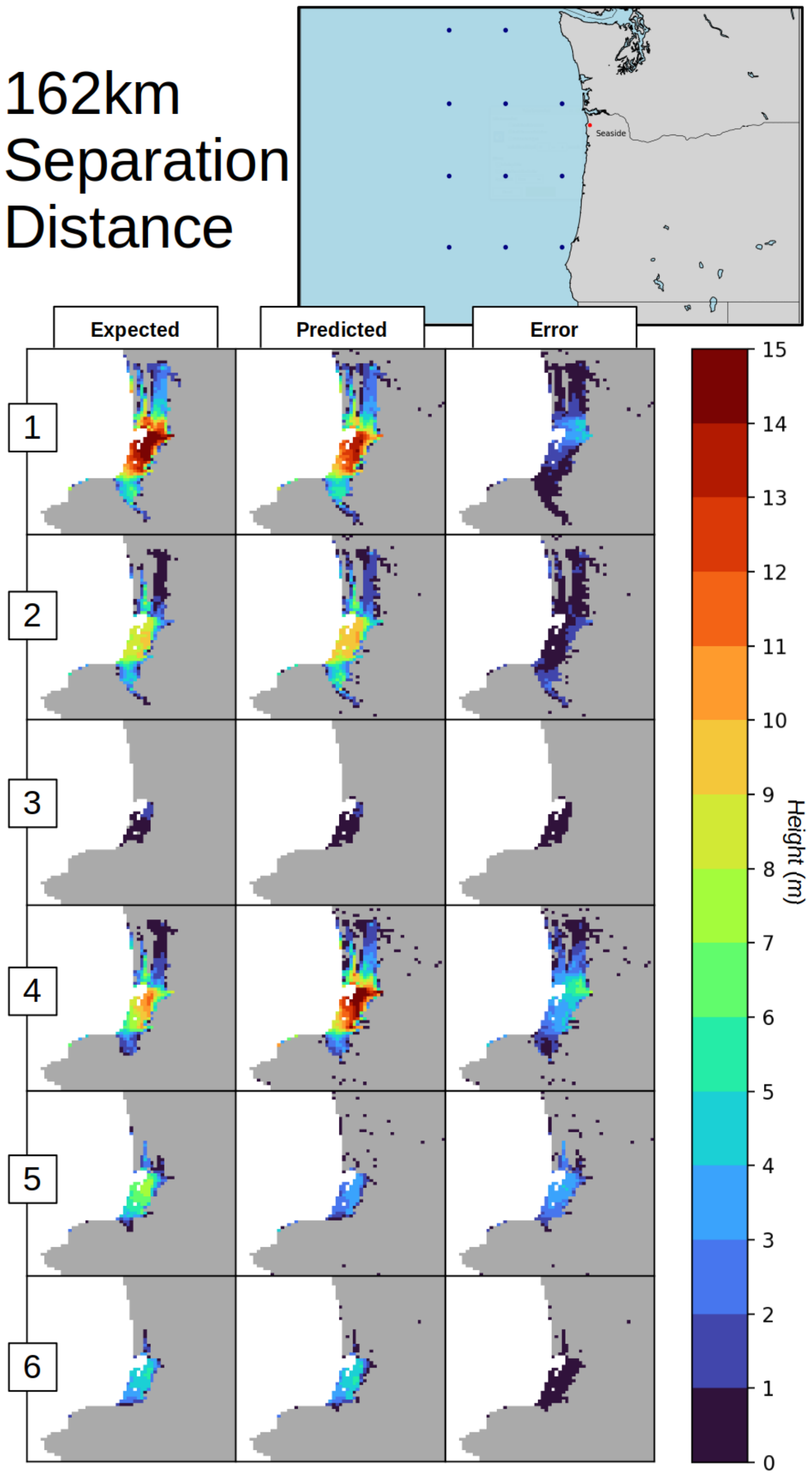


Figure 4.33: Results for six randomly selected events in the testing set using a hypothetical grid of buoys with 162 km separation distance, the arrangement with the highest error.

Overall, the results from the six randomly selected events using the existing buoys (Figure 4.31) seem acceptable. The error remains under 1 m for the most part with the exception of event #5 which exhibits error in the 1-2 m range. The results from the 18 km separation distance grid (Figure 4.32) show more consistency as the error for all events shown is below 1 m. We begin to see erroneous predictions when viewing the results from the 162 km separation distance grid (Figure 4.33). With the exception of two out of the six events shown, the remaining four events display error above 2 m, with one event reaching up to 8 m.

Since a random selection of events does not reveal the true accuracy of these models, is valuable to analyze the predictions with the highest absolute error (Equation 4.22) and highest percent error (Equation 4.24). The results can be seen in Figure 4.34.

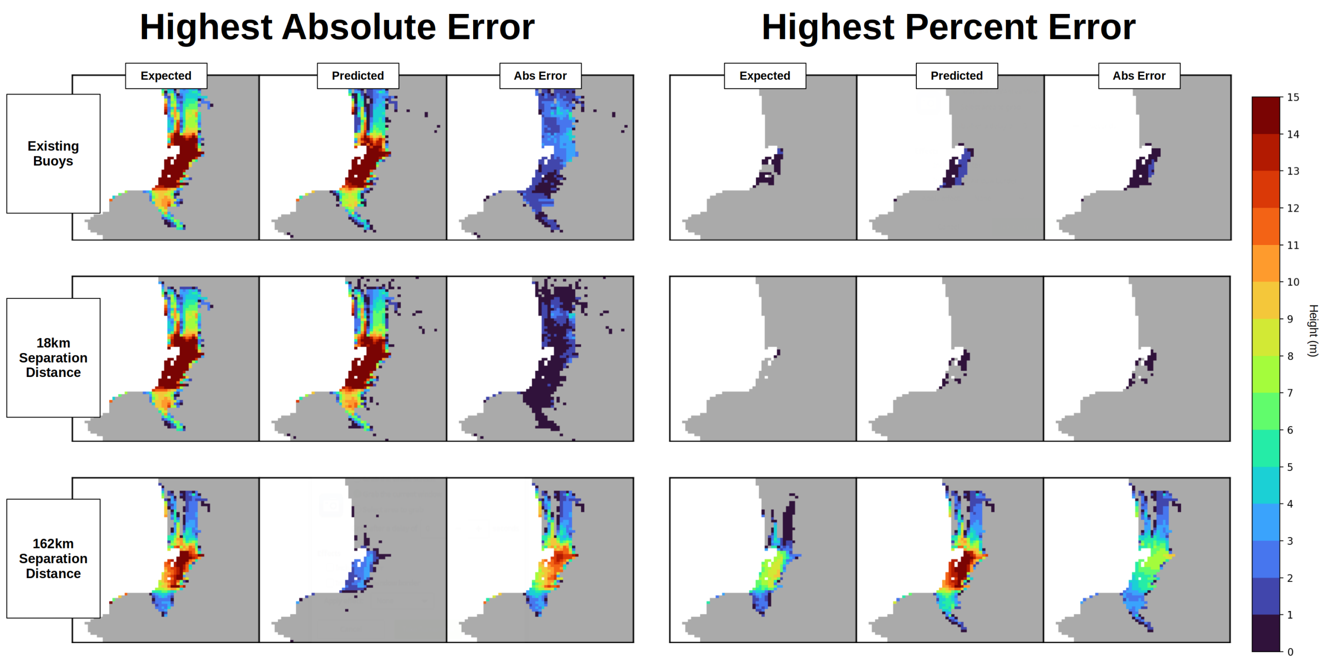


Figure 4.34: The events with the highest absolute error and the highest percent error for the models using existing buoys, an 18 km separation distance grid, and a 162 km separation distance grid.

It is clear from Figure 4.34 that both the highest absolute error prediction and the highest percent error prediction using the 162 km grid display significant deviation from the expected. This means given a particular event, there is a chance for a severely inaccurate prediction. Both the existing buoys and the tight arrangement of buoys perform moderately well in their respective worst case scenarios. However, the tight arrangement of buoys perform better in the worst case. This is in line with Figure 4.29 which shows a lower 90^{th} percentile for the tight grid of buoys compared to the existing buoys. It can also again be concluded that although the existing buoys perform similarly to a tight grid of buoys for a random selection of events, the severity of the most erroneous prediction decreases with a tight grid.

It is important to note that by the nature of a percent error formula, high percent error events will naturally tend towards those with relatively low inundation heights. Using the same logic, events with high absolute error tend towards those with high inundation heights. This tendency can be seen for both the worst case of the existing buoys and the 18 km grid. However, the 162 km grid does not follow this tendency because of the severity of the errors.

In the following tests, we aim to find an arrangement of the minimal amount of buoys required to achieve results equivalent to that of a dense network of tightly spaced buoys. The number of buoys contained in the grids tested above range from 10 for the loosest arrangement to over 800 for the tightest arrangement. However, not every time series from each buoy is equally utilized by the CNN when predicting inundation. The CNN naturally prioritizes buoys which contain more predictive power and as a result, only a few are necessary. The prioritization can be measured in terms of sensitivity, a measure of how much of an effect one buoy has on the accuracy.

To explore the sensitivity of the CNN with respect to its input buoy data, an occlusion test was performed (Makinoshima et al., 2021; Zeiler & Fergus, 2014; Nielsen & Voigt, 2018). It works by omitting one input parameter at a time and observing its effect on the accuracy. In this case, we first take the trained CNN for a particular grid configuration and run it without any modifications on the testing set to obtain the base average error. Then, the values of a selected buoy's time series are set to zero before giving it to the CNN to obtain a value of the modified average error. The ratio of the modified average error to the base average error is then calculated and scaled to range from one to two. A value of one would indicate that the particular buoy has no effect on the output of the prediction, while a value of two would indicate the buoy has strong influence on the prediction. This process is repeated for every buoy in the grid to obtain a map showing relative sensitivity. The sensitivity maps for the eight tightest grid arrangements are shown in Figure 4.35. To help aid in the visualization of the placement of the buoys, the peaks of the database-averaged tsunami waveform at the end of the buoy observational period were overlaid (13 min after the start of the simulation).

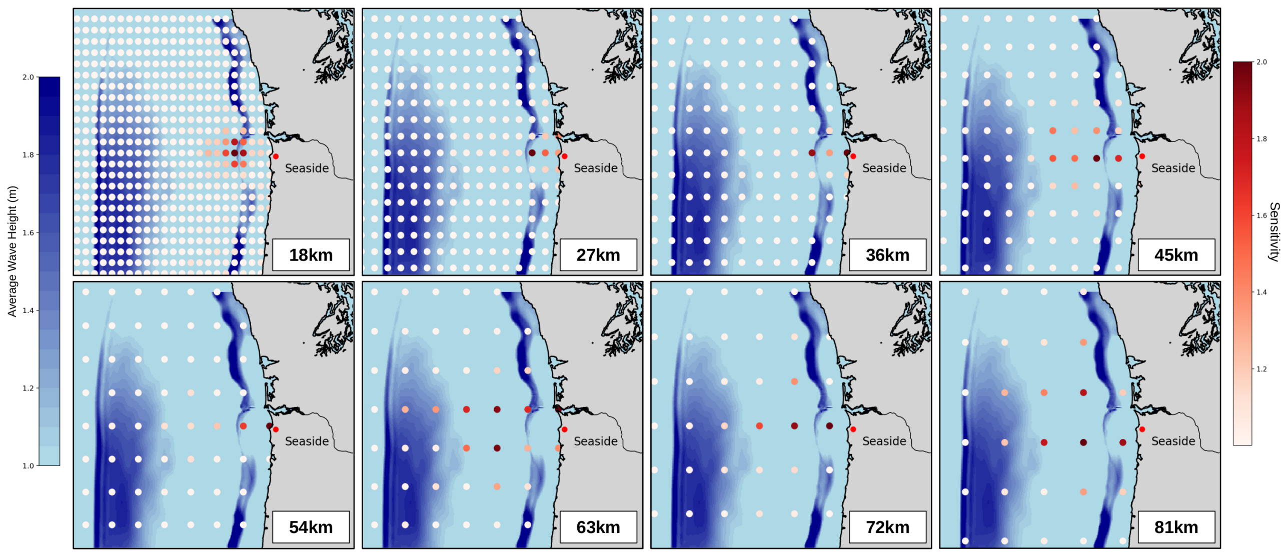


Figure 4.35: Buoy sensitivity tests for the eight tightest grids. Overlaid is the database-averaged waveform at the end of the buoy observational period (13 min). The CNN is most sensitive to buoys near the peak of the waveform at 13 min or buoys near the site of interest.

An interesting result can be seen in the upper left hand plot showing the sensitivity map with the tightest buoy spacing. It indicates that if given access to an extremely dense grid of wave height time series data, the CNN will prioritize buoys at the location where the wave is at its peak (at the time the buoys stop collecting data). This sensitive region generally persists with increasing buoy separation with the exception of 36 km and 54 km, where the most sensitive buoys are the ones positioned nearest the site of interest. This is an important result as it indicates the CNN can gain similar accuracy by either prioritizing near the site of interest, or near the peak of the waveform. The positioning of the buoys is also important. From buoy spacings of 27 km, 36 km 45 km, 54 km, and 72 km, the most sensitive buoys were ones in a line extending perpendicularly out from the coast near the site of interest. The two grid spacings which did not share this characteristic, 63 km and 81 km, split the sensitivity between two perpendicular lines adjacent to the site of interest.

Using this information, further testing was done to find a configuration which uses a minimal amount of buoys while still maintaining the accuracy of a tight grid of buoys. The decided upon configuration can be seen in Figure 4.36. It consists of four buoys, spaced 27 km apart, arranged in a line extending out to sea from the point of interest.

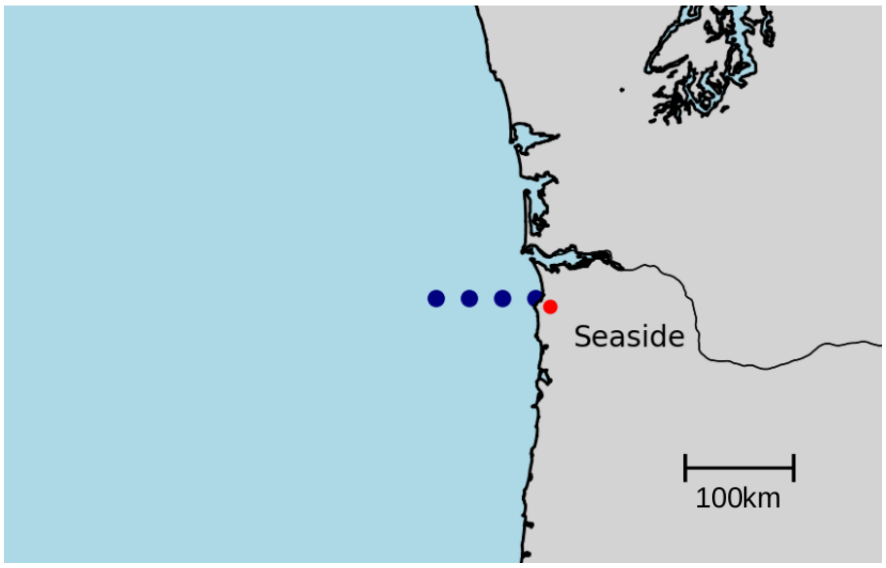


Figure 4.36: A minimal arrangement of buoys tested with aims to achieve similarly accurate results as a tight grid. Chosen based on sensitivity measurements (Figure 4.35) are four buoys spaced 27 km apart extending out to sea from the site of interest.

The results for the four buoy configuration in Figure 4.36 can be seen below in a layout similar to that of Figure 4.29. However, instead of the black horizontal lines representing existing buoys, they now represent the results of the four buoys arranged in a line extending out to sea.

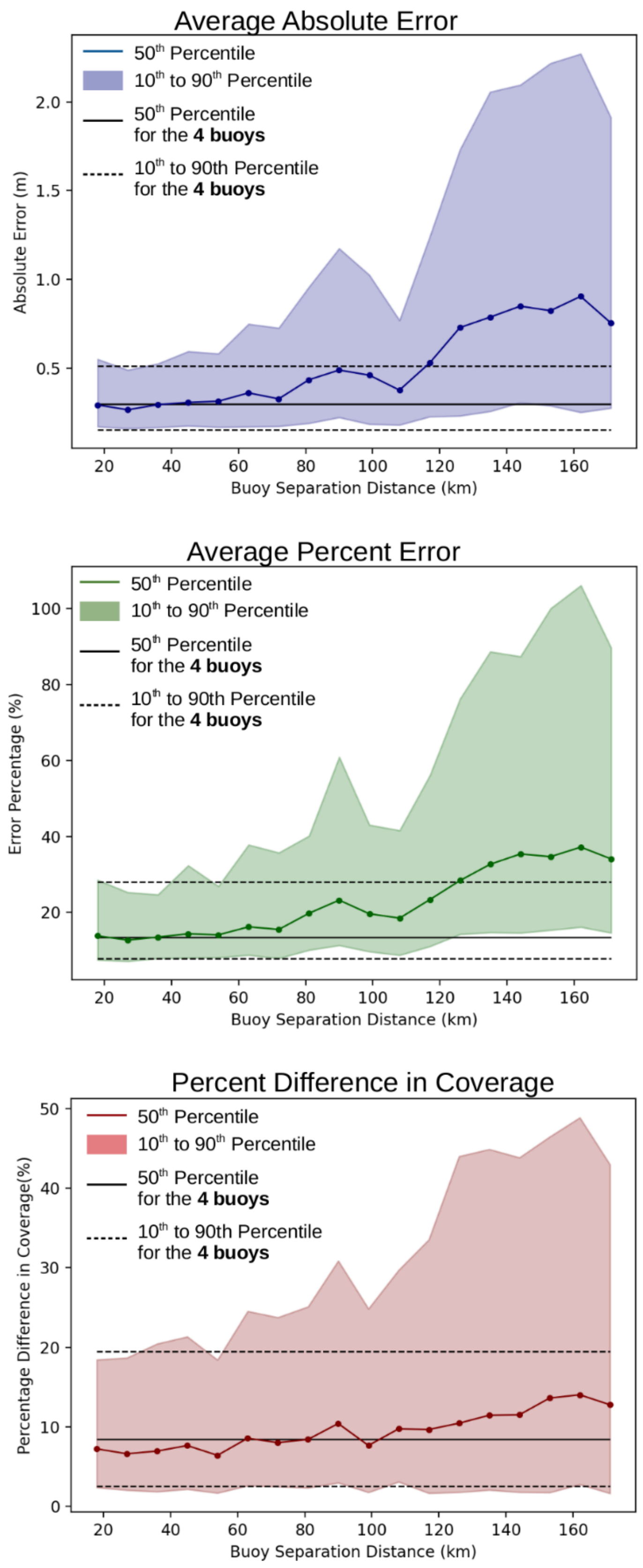


Figure 4.37: Plots identical to that of Figure 4.29 except with black lines now indicating results from the arrangement of four buoys (Figure 4.36). The four buoy arrangement achieves equally accurate results as the tightest grid spacings.

Results indicate that the accuracy using the four buoy arrangement for all three metrics are similar to that of a tight grid of buoys (separation distance <50 km). These results show that although a tight grid of buoys will produce inundation forecasts which are both accurate and consistent, the CNN does require all buoys in order to obtain an equally accurate forecast.

However, the findings here do not necessarily indicate that only four buoys are needed for any Cascadia Subduction Zone rupture scenario. Results are heavily confined by the locations and the magnitudes of the earthquake scenarios present in the database, namely earthquakes of magnitude ∼9 with epicenters very near the site of interest. Earthquakes outside these constraints may heavily affect which buoys are necessary. For example, if an earthquake with an epicenter near Southern Canada were to occur, the wave would travel mainly South until reaching the site of interest. In this case, a line of buoys extending East to West would not capture the incoming waveform, thus leading to an erroneous or even missed prediction. In addition, the four buoy arrangement may not produce similar results given a different buoy observational window. In this study, 13 min of buoy data was collected before input to the CNN. For shorter observational windows, buoys placed farther out to sea might be prioritized instead.

4.7 Conclusion

In this study, we presented two machine learning models (MLP and CNN) each trained with their own respective type of synthetically produced input data (GNSS ground deformation data and buoy wave height data) in order to forecast tsunami inundation due to a potential rupture of the Cascadia Subduction Zone for the town of Seaside, Oregon. Due to the short distance between the Cascadia Subduction Zone and the coast of Oregon, an early and accurate tsunami warning is crucial. Both methods introduce an alternative to existing direct non-linear forward modeling in real-time to obtain inundation forecasts, where they rely on rapid tsunami source estimations and significant computational resources to simulate nonlinear tsunami propagation. The MLP method uses GNSS ground deformation data as input and is similar to the method proposed by Fauzi & Mizutani (2020) where instead of using ground deformation data, authors use a low resolution grid of maximum wave heights generated by direct simulation to produce a high resolution inundation map. The CNN method uses buoy data as input and utilizes a neural network architecture similar to that of Makinoshima et al. (2021) where the difference lies in the type of forecast produced. Namely, the study here forecasts a map of the highest recorded tsunami flow depths similar to Fauzi & Mizutani (2020), while the study mentioned forecasts a time series waveform of a given point on land. The study here also produces earthquakes using a procedure similar to that of Goda et al. (2018), where correlations and target maximums were introduced to the slip map.

Tests using GNSS ground deformation data collected using the locations of currently operating stations were unable to produce accurate and reliable predictions. It was found that if an extremely large number of stations were used, an accurate prediction could be made. However, the hypothetical amount required would be impractical. It is currently unclear by how much of an effect an increase in the number of events in the database would have on the forecasting ability, and therefore more tests would be required to determine if this method is feasible.

Tests using time series buoy data from currently operating buoy stations were found to produce accurate inundation forecasts with a median error of $\sim$0.5 m. When using a hypothetical dense grid of buoys as input to the CNN, it was found to decrease the magnitude and frequency of large error predictions compared to the existing buoys. It was also found through a sensitivity analysis that only a small number of buoys were needed by the CNN to produce forecasts. Tests done indicate that a configuration of four strategically placed buoys were sufficient to obtain results as accurate as a dense grid of buoys for earthquakes localized near the Oregon coast for the city of Seaside.

The proposed CNN method using time series buoy data shows promise to produce fast and accurate inundation predictions. In addition, the method of determining the minimal number of necessary buoys through a sensitivity measure proved to be successful. In the future, we aim to expand the database to include a wider range of earthquake magnitudes and epicenter locations to more accurately cover the total possible rupture scenarios for the Cascadia Subduction Zone. The length of the buoy observational period will also be varied in future tests to determine its effect on accuracy, exploring the possibility of a continuous prediction based on the current length of the buoy time series.

References

Ammon, C. J., Ji, C., Thio, H.-K., Robinson, D., Ni, S., Hjorleifsdottir, V., … others (2005). Rupture process of the 2004 sumatra-andaman earthquake. *Science*, *308*(5725), 1133–1139.

Ando, M., Ishida, M., Hayashi, Y., & Mizuki, C. (2011). Interviews with survivors of tohoku earthquake provide insights into fatality rate. *Eos, Transactions American Geophysical Union*, *92*(46), 411–412.

Atwater, B. F., Musumi-Rokkaku, S., Satake, K., Tsuji, Y., & Yamaguchi, D. K. (2011). *The orphan tsunami of 1700: Japanese clues to a parent earthquake in north america.* University of Washington Press.

Atwater, B. F., Nelson, A. R., Clague, J. J., Carver, G. A., Yamaguchi, D. K., Bobrowsky, P. T., … others (1995). Summary of coastal geologic evidence for past great earthquakes at the cascadia subduction zone. *Earthquake spectra*, *11*(1), 1–18.

Bureau, U. S. C. (2022). *U.s. census website.* `https://www.census.gov/`. (date accessed: 2022-05-15)

Center, N. N. D. B. (2022). *Station 46089 - tillamook, or.* `https://www.ndbc.noaa.gov/station_page.php?station=46089`. (date accessed: 2022-04-07)

Center, N. N. G. D. (Ed.). (2003). *U.s. coastal relief model vol.8 - northwest pacific.* NOAA National Centers for Environmental Information. (date accessed: 2021-1-05) doi: 10.7289/V5H12ZXJ

Center, N. N. G. D. (Ed.). (2009). *Etopo1 1 arc-minute global relief model.* NOAA National Centers for Environmental Information. (date accessed: 2021-1-05) doi: 10.7289/V5C8276M

Ciregan, D., Meier, U., & Schmidhuber, J. (2012). Multi-column deep neural networks for image classification. In *2012 ieee conference on computer vision and pattern recognition* (pp. 3642–3649).

Clague, J. J. (1997). Evidence for large earthquakes at the cascadia subduction zone. *Reviews of Geophysics*, *35*(4), 439–460.

Delouis, B., Nocquet, J.-M., & Vallée, M. (2010). Slip distribution of the february 27, 2010 mw= 8.8 maule earthquake, central chile, from static and high-rate gps, insar, and broadband teleseismic data. *Geophysical Research Letters*, *37*(17).

Dominey-Howes, D., Dunbar, P., Varner, J., & Papathoma-Köhle, M. (2010). Estimating probable maximum loss from a cascadia tsunami. *Natural hazards*, *53*(1), 43–61.

Fauzi, A., & Mizutani, N. (2020). Machine learning algorithms for real-time tsunami inundation forecasting: a case study in nankai region. *Pure and Applied Geophysics*, *177*(3), 1437–1450.

Fiedorowicz, B., & Peterson, C. (2002). Tsunami deposit mapping at seaside, oregon, usa. *Geoenvironmental Mapping*, *1*, 630–648.

Fritz, H. M., Petroff, C. M., Catalán, P. A., Cienfuegos, R., Winckler, P., Kalligeris, N., ... others (2011). Field survey of the 27 february 2010 chile tsunami. *Pure and Applied Geophysics*, *168*(11), 1989–2010.

Garcia, C., & Delakis, M. (2004). Convolutional face finder: A neural architecture for fast and robust face detection. *IEEE Transactions on pattern analysis and machine intelligence*, *26*(11), 1408–1423.

Goda, K., Yasuda, T., Mai, P. M., Maruyama, T., & Mori, N. (2018). Tsunami simulations of mega-thrust earthquakes in the nankai–tonankai trough (japan) based on stochastic rupture scenarios. *Geological Society, London, Special Publications*, *456*(1), 55–74.

González, F., Geist, E. L., Jaffe, B., Kânoğlu, U., Mofjeld, H., Synolakis, C., ... others (2009). Probabilistic tsunami hazard assessment at seaside, oregon, for near-and far-field seismic sources. *Journal of Geophysical Research: Oceans*, *114*(C11).

Gusman, A. R., Tanioka, Y., MacInnes, B. T., & Tsushima, H. (2014). A methodology for near-field tsunami inundation forecasting: Application to the 2011 tohoku tsunami. *Journal of Geophysical Research: Solid Earth*, *119*(11), 8186–8206.

Haidvogel, D. B., Arango, H., Budgell, W. P., Cornuelle, B. D., Curchitser, E., Di Lorenzo, E., ... others (2008). Ocean forecasting in terrain-following coordinates: Formulation and skill assessment of the regional ocean modeling system. *Journal of Computational Physics*, *227*(7), 3595–3624.

Heaton, J. (2008). *Introduction to neural networks with java.* Heaton Research, Inc.

Heaton, T. H., & Hartzell, S. H. (1987). Earthquake hazards on the cascadia subduction zone. *Science*, *236*(4798), 162–168.

Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. *Neural networks*, *4*(2), 251–257.

Imamura, F., & Anawat, S. (2011). Damage due to the 2011 tohoku earthquake tsunami and its lessons for future mitigation. In *Proceedings of the international symposium on engineering lessons learned from the* (pp. 1–4).

Ishii, M., Shearer, P. M., Houston, H., & Vidale, J. E. (2005). Extent, duration and speed of the 2004 sumatra–andaman earthquake imaged by the hi-net array. *Nature*, *435*(7044), 933–936.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kubota, T., Suzuki, W., Nakamura, T., Chikasada, N. Y., Aoi, S., Takahashi, N., & Hino, R. (2018). Tsunami source inversion using time-derivative waveform of offshore pressure records to reduce effects of non-tsunami components. *Geophysical Journal International*, *215*(2), 1200–1214.

Lander, J. F., Lockridge, P. A., & Kozuch, M. J. (1993). *Tsunamis affecting the west coast of the united states, 1806-1992* (No. 29). US Department of Commerce, National Oceanic and Atmospheric Administration . . . .

Lawrence, S., Giles, C. L., Tsoi, A. C., & Back, A. D. (1997). Face recognition: A convolutional neural-network approach. *IEEE transactions on neural networks*, *8*(1), 98–113.

Lay, T., Ammon, C. J., Kanamori, H., Koper, K., Sufri, O., & Hutko, A. (2010). Teleseismic inversion for rupture process of the 27 february 2010 chile (mw 8.8) earthquake. *Geophysical Research Letters*, *37*(13).

Lay, T., Ammon, C. J., Kanamori, H., Xue, L., & Kim, M. J. (2011). Possible large near-trench slip during the 2011 m w 9.0 off the pacific coast of tohoku earthquake. *Earth, planets and space*, *63*(7), 32.

Lay, T., Kanamori, H., Ammon, C. J., Nettles, M., Ward, S. N., Aster, R. C., . . . others (2005). The great sumatra-andaman earthquake of 26 december 2004. *Science*, *308*(5725), 1127–1133.

Lin, M., Chen, Q., & Yan, S. (2013). Network in network. *arXiv preprint arXiv:1312.4400*.

Lo, S.-C. B., Chan, H.-P., Lin, J.-S., Li, H., Freedman, M. T., & Mun, S. K. (1995). Artificial convolution neural network for medical image pattern recognition. *Neural networks*, *8*(7-8), 1201–1214.

Løvholt, F., Pedersen, G., Bazin, S., Kühn, D., Bredesen, R. E., & Harbitz, C. (2012). Stochastic analysis of tsunami runup due to heterogeneous coseismic slip and dispersion. *Journal of Geophysical Research: Oceans*, *117*(C3).

Mai, P. M., Schorlemmer, D., Page, M., Ampuero, J.-P., Asano, K., Causse, M., ... others (2016). The earthquake-source inversion validation (siv) project. *Seismological Research Letters*, *87*(3), 690–708.

Makinoshima, F., Oishi, Y., Yamazaki, T., Furumura, T., & Imamura, F. (2021). Early forecasting of tsunami inundation from tsunami and geodetic observation data with convolutional neural networks. *Nature communications*, *12*(1), 1–10.

Mori, N., & Takahashi, T. (2012). The 2011 tohoku earthquake tsunami joint survey group (2012) nationwide post event survey and analysis of the 2011 tohoku earthquake tsunami. *Coastal Engineering Journal*, *54*(1), 1250001.

Mueller, C., Power, W., Fraser, S., & Wang, X. (2015). Effects of rupture complexity on local tsunami inundation: Implications for probabilistic tsunami hazard assessment by example. *Journal of Geophysical Research: Solid Earth*, *120*(1), 488–502.

Mulia, I. E., Gusman, A. R., & Satake, K. (2018). Alternative to non-linear model for simulating tsunami inundation in real-time. *Geophysical Journal International*, *214*(3), 2002–2013.

Mulia, I. E., Gusman, A. R., & Satake, K. (2020). Applying a deep learning algorithm to tsunami inundation database of megathrust earthquakes. *Journal of Geophysical Research: Solid Earth*, *125*(9), e2020JB019690.

Musa, A., Watanabe, O., Matsuoka, H., Hokari, H., Inoue, T., Murashima, Y., ... Kobayashi, H. (2018). Real-time tsunami inundation forecast system for tsunami disaster prevention and mitigation. *The Journal of Supercomputing*, *74*(7), 3093–3113.

Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Icml*.

Nielsen, A. A., & Voigt, C. A. (2018). Deep learning to predict the lab-of-origin of engineered dna. *Nature communications*, *9*(1), 1–10.

Oishi, Y., Imamura, F., & Sugawara, D. (2015). Near-field tsunami inundation forecast using the parallel tunami-n2 model: Application to the 2011 tohoku-oki earthquake combined with source inversions. *Geophysical Research Letters*, *42*(4), 1083–1091.

Okada, Y. (1985). Surface deformation due to shear and tensile faults in a half-space. *Bulletin of the seismological society of America*, *75*(4), 1135–1154.

Paris, R., Wassmer, P., Sartohadi, J., Lavigne, F., Barthomeuf, B., Desgages, E., ... others (2009). Tsunamis as geomorphic crises: lessons from the december 26, 2004 tsunami in lhok nga, west banda aceh (sumatra, indonesia). *Geomorphology*, *104*(1-2), 59–72.

Park, H., Alam, M. S., Cox, D. T., Barbosa, A. R., & van de Lindt, J. W. (2019). Probabilistic seismic and tsunami damage analysis (pstda) of the cascadia subduction zone applied to seaside, oregon. *International Journal of Disaster Risk Reduction*, *35*, 101076.

Park, H., & Cox, D. T. (2016). Probabilistic assessment of near-field tsunami hazards: Inundation depth, velocity, momentum flux, arrival time, and duration applied to seaside, oregon. *Coastal Engineering*, *117*, 79–96.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., ... Lerer, A. (2017). Automatic differentiation in pytorch.

Peterson, C. D., Cruikshank, K. M., Jol, H. M., & Schlichting, R. B. (2008). Minimum runup heights of paleotsunami from evidence of sand ridge overtopping at cannon beach, oregon, central cascadia margin, usa. *Journal of Sedimentary Research*, *78*(6), 390–409.

Priest, G. R., Stimely, L. L., Wood, N. J., Madin, I. P., & Watzig, R. J. (2016). Beat-the-wave evacuation mapping for tsunami hazards in seaside, oregon, usa. *Natural Hazards*, *80*(2), 1031–1056.

Röbke, B., & Vött, A. (2017). The tsunami phenomenon. *Progress in Oceanography*, *159*, 296–322.

Rosenblatt, F. (1957). *The perceptron, a perceiving and recognizing automaton project para.* Cornell Aeronautical Laboratory.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., ... others (2015). Imagenet large scale visual recognition challenge. *International journal of computer vision*, *115*(3), 211–252.

Satake, K. (2014). The 2011 tohoku, japan, earthquake and tsunami. *Extreme Natural Hazards, Disaster Risks and Societal Implications*, *1*, 310–321.

Satake, K., Fujii, Y., Harada, T., & Namegaya, Y. (2013). Time and space distribution of coseismic slip of the 2011 tohoku earthquake as inferred from tsunami waveform data. *Bulletin of the seismological society of America*, *103*(2B), 1473–1492.

Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., & LeCun, Y. (2013). Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*.

Simons, M., Minson, S. E., Sladen, A., Ortega, F., Jiang, J., Owen, S. E., ... others (2011). The 2011 magnitude 9.0 tohoku-oki earthquake: Mosaicking the megathrust from seconds to centuries. *science*, *332*(6036), 1421–1425.

Song, Y. T. (2007). Detecting tsunami genesis and scales directly from coastal gps stations. *Geophysical Research Letters*, *34*(19).

Song, Y. T., Fukumori, I., Shum, C., & Yi, Y. (2012). Merging tsunamis of the 2011 tohoku-oki earthquake detected over the open ocean. *Geophysical Research Letters*, *39*(5).

Song, Y. T., Mohtat, A., & Yim, S. C. (2017). New insights on tsunami genesis and energy source. *Journal of Geophysical Research: Oceans*, *122*(5), 4238–4256.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, *15*(1), 1929–1958.

Stein, S., & Okal, E. A. (2005). Speed and size of the sumatra earthquake. *Nature*, *434*(7033), 581–582.

Stern, R. J. (2002). Subduction zones. *Reviews of geophysics*, *40*(4), 3–1.

Subarya, C., Chlieh, M., Prawirodirdjo, L., Avouac, J.-P., Bock, Y., Sieh, K., … McCaffrey, R. (2006). Plate-boundary deformation associated with the great sumatra–andaman earthquake. *Nature*, *440*(7080), 46–51.

Survey, N. G. (2022). *The noaa cors network (ncn).* `https://geodesy.noaa.gov/CORS/`. (date accessed: 2022-04-02)

Synolakis, C. E. (1987). The runup of solitary waves. *Journal of Fluid Mechanics*, *185*, 523–545.

Tang, L., Titov, V., Wei, Y., Mofjeld, H., Spillane, M., Arcas, D., … Newman, J. (2008). Tsunami forecast analysis for the may 2006 tonga tsunami. *Journal of Geophysical Research: Oceans*, *113*(C12).

Team, N. G. S. C. (2013). *Gnss receiver - port angeles, wa.* `https://www.ngs.noaa.gov/cgi-cors/corsage.prl?site=PTAA`. (date accessed: 2022-04-02)

Toledo-Marín, J. Q., Fox, G., Sluka, J. P., & Glazier, J. A. (2021). Deep learning approaches to surrogates for solving the diffusion equation for mechanistic real-world simulations. *Frontiers in Physiology*, *12*.

Tsushima, H., Hino, R., Ohta, Y., Iinuma, T., & Miura, S. (2014). tfish/rapid: Rapid improvement of near-field tsunami forecasting based on offshore tsunami data by incorporating onshore gnss data. *Geophysical Research Letters*, *41*(10), 3390–3397.

Tsushima, H., Hino, R., Tanioka, Y., Imamura, F., & Fujimoto, H. (2012). Tsunami waveform inversion incorporating permanent seafloor deformation and its application to tsunami forecasting. *Journal of Geophysical Research: Solid Earth*, *117*(B3).

Tsushima, H., Hirata, K., Hayashi, Y., Tanioka, Y., Kimura, K., Sakai, S., … Maeda, K. (2011). Near-field tsunami forecasting using offshore tsunami data from the 2011 off the pacific coast of tohoku earthquake. *Earth, planets and space*, *63*(7), 821–826.

Wang, J., Ward, S. N., & Xiao, L. (2015). Numerical modelling of rapid, flow-like landslides across 3-d terrains: a tsunami squares approach to el picacho landslide, el salvador, september 19, 1982. *Geophysical Journal International*, *201*(3), 1534–1544.

Wang, J., Ward, S. N., & Xiao, L. (2019). Tsunami squares modelling of the 2015 june 24 hongyanzi landslide generated river tsunami in three gorges reservoir, china. *Geophysical Journal International*, *216*(1), 287–295.

Ward, S. N., & Day, S. (2006). Particulate kinematic simulations of debris avalanches: interpretation of deposits and landslide seismic signals of mount saint helens, 1980 may 18. *Geophysical Journal International*, *167*(2), 991–1004.

Ward, S. N., & Day, S. (2008). Tsunami balls: a granular approach to tsunami runup and inundation. *Communications in computational Physics*, *3*(1), 222–249.

Ward, S. N., & Day, S. (2010). The 1958 lituya bay landslide and tsunami—a tsunami ball approach. *Journal of Earthquake and Tsunami*, *4*(04), 285–319.

Ward, S. N., & Day, S. (2011). The 1963 landslide and flood at vaiont reservoir italy. a tsunami ball simulation. *Italian Journal of Geosciences*, *130*(1), 16–26.

Wilson, J. M., Schultz, K. W., Grzan, D., Rundle, J. B., Ward, S. N., Bhaskar, R., … Kaushal, H. (2020). Tsunami squares simulation of megathrust-generated waves: Application to the 2011 tohoku tsunami. *Progress in Disaster Science*, *5*, 100063.

Xiao, L., Ward, S. N., & Wang, J. (2015). Tsunami squares approach to landslide-generated waves: application to gongjiafang landslide, three gorges reservoir, china. *Pure and Applied Geophysics*, *172*(12), 3639–3654.

Zeiler, M. D., & Fergus, R. (2014). Visualizing and understanding convolutional networks. In *European conference on computer vision* (pp. 818–833).