Pure Appl. Geophys. 178 (2021), 5129–5153

https://doi.org/10.1007/s00024-021-02841-9

Pure and Applied Geophysics

Comparison of Machine Learning Approaches for Tsunami Forecasting from Sparse Observations

Christopher M. Liu,[1] Donsub Rim,[2] Robert Baraldi,[1] and Randall J. LeVeque[1]

Abstract—We have explored various different machine learning (ML) approaches for forecasting tsunami amplitudes at a set of forecast points, based on hypothetical short-time observations at one or more observation points. As a case study, we chose an observation point near the entrance of the Strait of Juan de Fuca, and two forecast points in the Salish Sea, one in Discovery Bay and the other in Admiralty Inlet, the waterway leading to southern Puget Sound. One ML approach considered is to train a support vector machine to predict the maximum amplitude at the forecast points. We also explored the use of two deep convolutional neural networks, a denoising autoencoder and a variational autoencoder to predict the full time series at the forecast points. These latter approaches also provide an estimate of the uncertainty in the predictions. As training data we use a subset of the 1300 synthetic CSZ earthquakes generated in the work of Melgar et al. (J Geophys Res Solid Earth 121:6658-6674, 2016b), reserving some as test data. As additional tests, the trained ML models have also been applied to other hypothetical CSZ earthquakes produced by very different approaches, such as the "L1 event" from the work of Witter et al. (Geosphere 9(6):1783-1803, 2013) that is used in the generation of tsunami inundation maps in Washington State. The ML models are capable of providing very good predictions from short duration observations, even when truncated before the first wave peak has reached the observation point.

Keywords: Tsunami forecasting, machine learning, GeoClaw.

1. Introduction

Real-time tsunami forecasts must be generated as quickly as possible using incomplete data. A variety of different data streams can be useful, e.g. seismic waveforms, GNSS data, and/or direct tsunami observations. In this paper we focus on tsunami surface elevation data, as might be inferred from ocean bottom pressure sensors, or measured directly by tide gauges or high-frequency radar observations, for example; see e.g. (Bernard & Titov, 2015; Grilli et al., 2016; Melgar et al., 2016a; Mulia & Satake, 2020). The ideas we explore could potentially be applied to other data streams as well.

We focus on one case study, forecasting tsunami amplitudes at locations in the Salish Sea based on hypothetical sea surface elevation observations near the entrance of the Strait of Juan de Fuca (SJdF). The Salish Sea includes Puget Sound in Washington State and the Strait of Georgia in Canada, with a large coastal population in Seattle, Tacoma, Vancouver, and numerous smaller communities. Crustal faults running across the Salish Sea could generate local tsunamis, but in this paper we focus on tsunamis entering the Strait from the Pacific Ocean, in particular from a megathurst earthquake on the Cascadia Subduction Zone (CSZ). The Strait is essentially the only path for tsunamis to reach the highly-populated portions of the Salish Sea from the Pacific, and so we hypothesize that direct observation of the tsunami near the entrance of the Strait should be adequate to predict tsunami amplitudes within the Salish Sea. Moreover, it takes 1–2 h for the tsunami to travel from the entrance of the Strait to many locations of interest, potentially providing time for useful forecasting even from a CSZ event.

This research was supported in part by the Air Force Center of Excellence under Award Number FA9550-17-1-0195 and AFOSR MURI Award Number FA9550-15-1-0038, the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Department of Energy Computational Science Graduate Fellowship under Award Number DE-FG02-97ER25308, and by Tohoku University, Sendai, Japan.

[1] Department of Applied Mathematics, University of Washington, Seattle, WA 98195, USA. E-mail: cmhl@uw.edu; rbaraldi@uw.edu; rjl@uw.edu

[2] Department of Mathematics and Statistics, Washington University in St. Louis, St. Louis, MO 63130, USA. E-mail: rim@wustl.edu

Hence we wish to use observations over a short time period at one location to predict maximum amplitudes over several hours at several other locations. We explore the use of machine learning (ML) approaches to facilitate this. We train a model using hundreds of hypothetical earthquake realizations, for which full time series of surface elevations are generated at both the observation point and the forecast points using the GeoClaw tsunami modeling software. The input for the ML model is the observation over some short time period (the observation window T_{ow}, 30 or 60 min for our tests) at the observation point. The desired output, at the set of forecast locations, is either the full time series over a longer time period (the forecast window T_{fw}, 5 h in our case), or simply the maximum predicted amplitude over this time period.

Many different methods are already in use for operational tsunami warning; see (Bernard & Titov, 2015) or (Mulia & Satake, 2020) for recent surveys. Recently a number of research groups have also explored the use of ML to develop new methodologies to rapidly predict time series at forecast points (as we consider) and/or maximum inundation estimates over a specified coastal region based on truncated time series at one or more observation points. One approach is to compute a database of synthetic model results and then to use these to train an algorithm to select the model from the database that is thought to give the best agreement with a new observation. The time series at a coastal location (and/or inundation map) from that particular model is then used as the forecast for the new event. This approach was used, for example, by Mulia et al. (2016) and Fauzi and Mizutani (2020). The latter authors also compared it with an approach that is more similar to the one we adopt, in which the ML algorithm using a shallow neural network directly generates a new forecast rather than selecting a precomputed model from a database. Mulia et al. (2020) also considered a related approach, in which running a low-resolution tsunami model in real time during an event gives the input for a deep neural network trained to produce a new high-resolution forecast. The methods proposed in these recent papers depend on a rapid and reliable source inversion to generate initial conditions for a low-resolution model run.

In this paper we consider approaches that do not require source inversion or running any tsunami model in real-time during an event, and that predict the maximum amplitude and/or the full time series at selected forecast points, based on a truncated time series at a single observation point. In theory these approaches could be extended to include multiple observation points, but for waves entering the Salish Sea we show that a single observation point is sufficient to obtain good results.

In the first approach we consider, we use the `tsfresh` software (Christ et al., 2016, 2018) to extract features from the input time series and then feed these features into ML algorithms implemented in `scikit-learn` (Pedregosa et al., 2011). The goal is to predict the *maximum* tsunami amplitude at each forecast location, without predicting the full time series. This approach is discussed further in Sect. 4. We experimented with various widely used models, and settled on Support Vector Machines (SVMs) with and without feature extraction. In the latter case, we trained the SVM on the entire time-series. SVMs provide a relatively simple model to contrast against more complicated models described below. In addition, they may perform better when data lacks the capacity to fully train a complicated neural network. Finally, use of `tsfresh` may reveal what particular features of the dataset is particularly relevant; these details would be lost in the parameterization of our more complicated models.

The other approaches we consider are based on deep neural networks and autoencoders, and are able to predict a *full time series* at the forecast locations. This provides more forecast information than the maximum alone, but does not provide interpretable information on what features of the input signal are most important in making the predictions. We comment further on this Sect. 8.

We are also interested in providing some estimate of the uncertainty in each forecast produced. We consider two different approaches to this in the context of autoencoders, as explained in Sects. 5 and 6.

The case we focus on in this paper is a relatively simple case of tsunami forecasting, because observations (over a sufficiently long time period) at a single point are perhaps sufficient to determine the tsunami amplitude throughout the Salish Sea, e.g. by

performing a tsunami simulation over the full Salish Sea using the full time series of observations at the Strait entrance as boundary conditions. The ML problem is still nontrivial, however, and captures two primary challenges that must be met to make this approach feasible for rapid real-time forecasting following an earthquake: (a) to use ML to create a model that can quickly make these forecasts without the need for new tsunami modeling based on the observations, and (b) to make the forecast using truncated observations over as short a time period as possible.

The results of this study are sufficiently promising that we plan to extend this work to more challenging tsunami forecasting problems. For example, predicting coastal tsunami amplitudes based on observations at one or more DART buoys would be more difficult since sparse observations at a few points in the deep ocean do not constrain the behavior at coastal locations to nearly the same degree as is possible within the Salish Sea using data from the entrance to the Strait. We discuss other possible directions for future research in Sect. 8.

In this work we have used the GeoClaw software for the tsunami simulations, which is distributed as part of the open source package Clawpack (Clawpack Development Team, 2020). This code has been validated for tsunami modeling by the U.S. National Tsunami Hazard Mitigation Program (NTHMP) after conducting multiple benchmark tests as part of an NTHMP benchmarking workshop (González, et al., 2011; NTHMP, 2011), and is currently being used to produce inundation maps for much of Puget Sound and the outer coast of Washington State.

Note that we discuss the Karhunen-Loève series in the context of generating random earthquakes and the Kullback-Leibler divergence in the context of the variational autoencoder. Both are abbreviated KL in different contexts.

2. *Generation of Synthetic Data*

In order to train an ML algorithm we need a large set of training data, in our case gauge time series records for many synthetic tsunamis entering the Strait of Juan de Fuca. Since there has not been a significant CSZ event since 1700 there is no data available from real events. Even in other parts of the world where major subduction zone earthquakes are more common, there are only a small handful of historical events. Thus we must generate synthetic data by performing numerical tsunami simulations based on hypothetical ground motions that are generated in a manner that is geophysically reasonable. For this paper we have used a set of 1300 synthetic CSZ events that were generated by Melgar et al. (2016b) and archived at Melgar (2016), and that range in magnitude from Mw 7.8 to 9.3. These were originally used for testing the use of GNSS data in the context of earthquake early warning, but the ground motions can also be used as initial data for tsunami simulations. These realizations were generated using a Karhunen-Loève (KL) expansion as proposed by LeVeque et al. (2016), which provides a simple way to generate a large number of random lognormal slip distributions with desired statistical properties. The Slab 1.0 geometry for the CSZ (Hayes et al., 2012) was discretized using 963 triangular subfaults, as shown in Fig. 1. The KL expansion is based on the eigenvectors of the desired 963×963 covariance matrix that relates slip on each pair of subfaults, and was specified using a Von Karman correlation function with Hurst exponent $H = 0.75$ and correlation lengths based on the effective length and width of the fault, which was reduced from the full fault geometry based on the target magnitude. In addition, kinematic rupture parameters were also added so that the rupture was initiated at a randomly selected hypocenter and the rupture propagated outwards over the course of a few minutes. Full details can be found in the papers cited above.

Each random realization corresponds to a collection of slip values (meters of slip) on each triangular subfault, together with the rupture time and duration. From these the time-dependent seafloor deformation is computed as suggested by Comninou and Dundurs (1975), a variant of the model proposed by Okada (1985) for rectangular subfaults and commonly used in tsunami modeling. Both triangular and rectangular subfault versions are implemented in Python as part of the GeoClaw software. Figure 1 shows the slip distribution from one sample realization #1127, a Mw 8.9 event that initiates near the southern end of CSZ

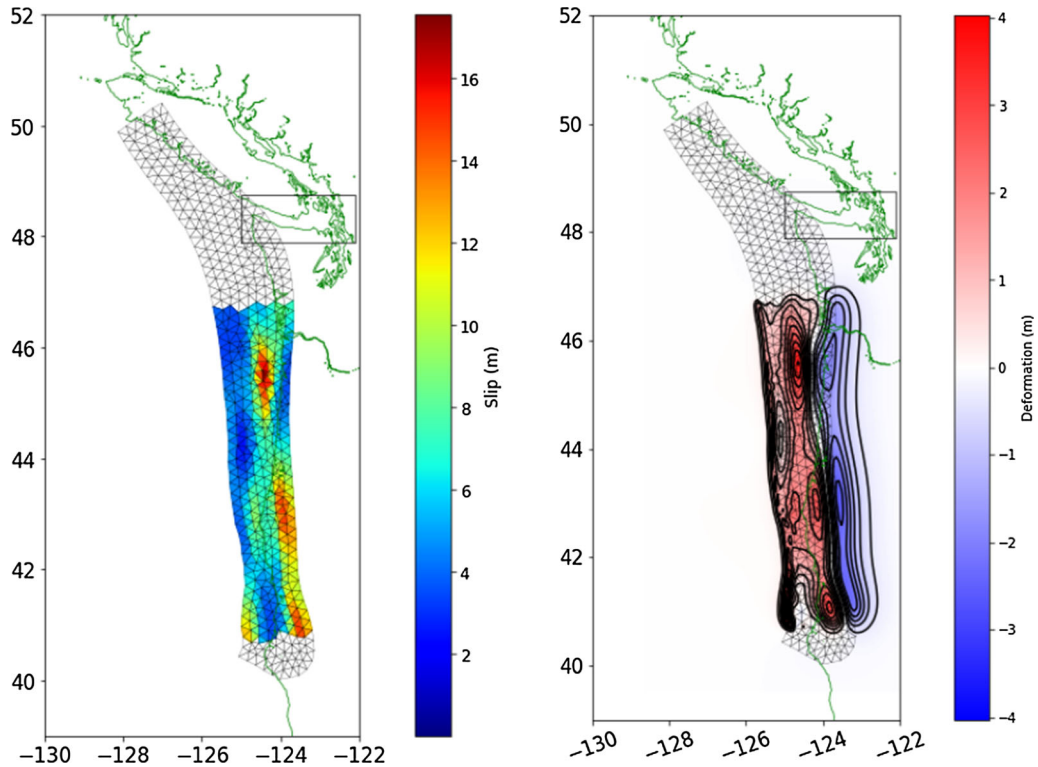

Figure 1
Left: Surface projection of the triangular subfaults of Slab 1.0 model of CSZ together with slip from a sample realization #1127, with Mw 8.8. Right: Final seafloor deformation for realization #1127 as computed with the Okada model. In each figure, the black rectangle shows the SJdF region that is shown in more detail in Fig. 2

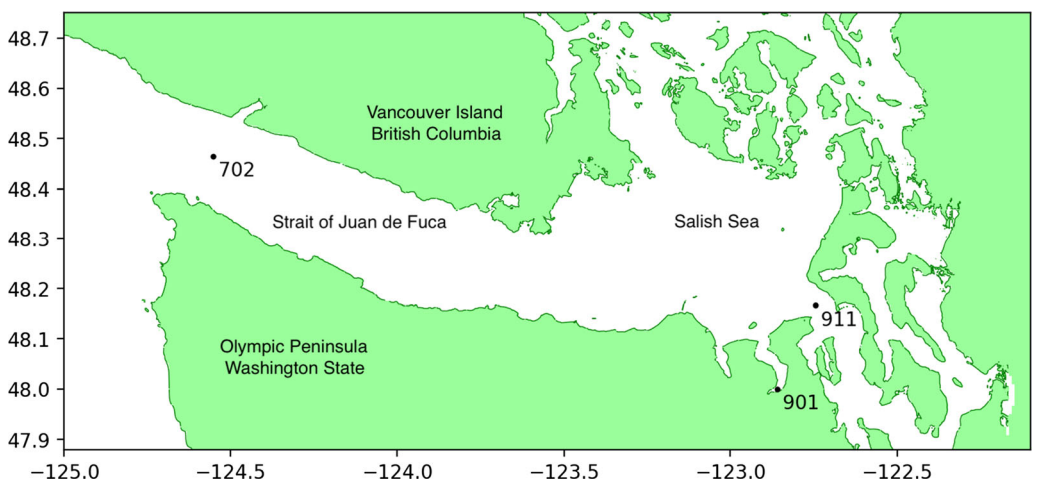

Figure 2
The Strait of Juan de Fuca (SJdF) region with gauge locations. Gauge 702 is the observation gauge, Gauge 901 is at the terminus of Discovery Bay, and Gauge 911 is in Admiralty Inlet. The Salish Sea extends both south and north of the region shown. Seattle and Tacoma lie to the south, while Vancouver, BC is to the north

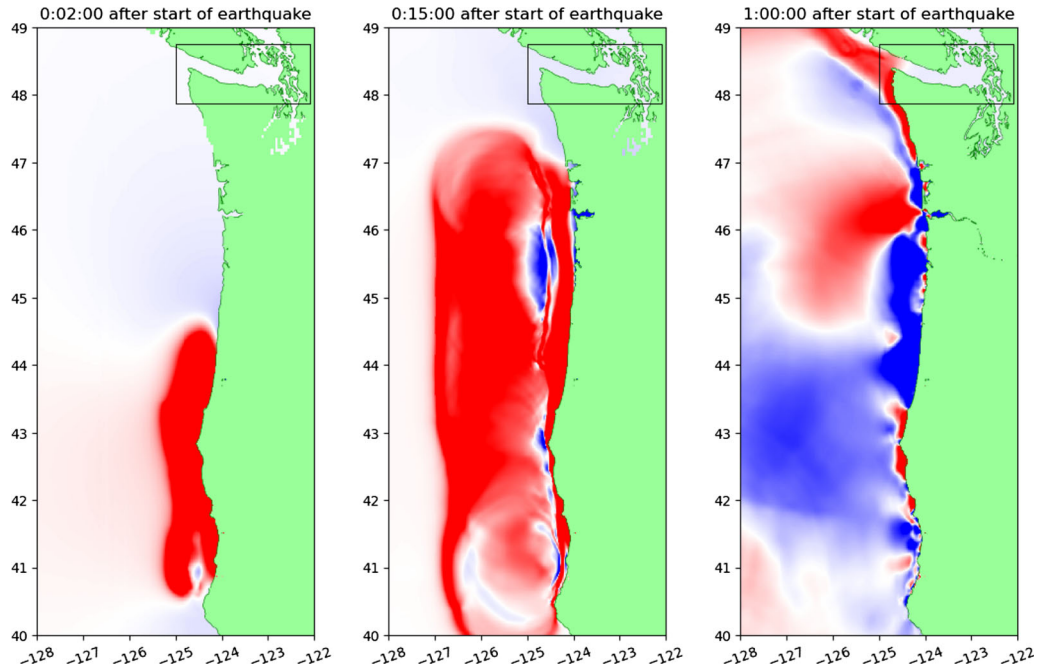

Figure 3
Tsunami from Realization #1127 at times 0:02, 0:15 and 1:00. Colors show surface elevation and saturate at − 0.5 m (blue) and + 0.5 m (red). Note that at time 0:02 the sea floor deformation is still in progress. For this realization, the earthquake ruptures from south to north with a total duration of about 4 min. The SJdF region is indicated by the rectangle.

and ruptures northward over roughly 4 min. We use the time-dependent rupture in our tsunami modeling of these events, although the recent work of Williamson et al. (2019) shows that this is expected to have little effect in the context we are studying, relative to the common practice of using an instantaneous displacement of the entire seafloor to the final displacement. Moreover, although we always use these time-dependent deformations in our training data, we also tested the resulting models using some other tsunami source models that consist of instantaneous displacement.

The tsunami generation and resulting propagation is then modeled by the nonlinear shallow water equations, using the high-resolution finite volume methods and adaptive mesh refinement implemented in Fortran with OpenMP in GeoClaw. This efficient code enables the solution of hundreds or thousands of tsunami simulations, and has been used with large numbers of realizations in other contexts such as probabilistic tsunami hazard assessment (PTHA); see for example (Crempien et al., 2020; Williamson et al., 2020).

Synthetic gauges record the water elevation for each realization at a set of points in the computational domain. Figure 2 shows the locations of the gauges used in this work. Gauge 702 is located near the entrance of the Strait, and we assume that observations are available at this “observation point”. The other gauges are located at hypothetical “forecast points”, where we hope to make rapid forecasts of the tsunami amplitude. Gauge 901 is located deep in Discovery Bay, where field work has uncovered tsunami deposits from several past CSZ events; see e.g. (Garrison-Laney, 2017; Williams et al., 2005). Gauge 911 is located in the middle of Admiralty Inlet, the passage between Port Townsend and Whidbey Island that leads to southern Puget Sound.

Each tsunami simulation was run for 6 h of simulated time, with adaptive refinement used to allow refinement to varying resolutions in the computational domain depending on the resolution needed to

capture the waves of interest. In particular, 30 arc-second resolution was allowed in the Cascadia source region and through the Strait and Admiralty Inlet, while finer 2 arcsecond grids were allowed in Discovery Bay to better capture inundation of this region. The simulations were run on an "ocean at rest" so the effects of tides are not included. Real observation data would have to be de-tided before providing as input to a machine learning model trained in this manner.

Figure 3 shows three frames from the tsunami simulation for the sample realization #1127, at 2 min, 15 min , and 1 h after initiation of the kinematic rupture that evolved over roughtly 4 min. For this particular realization, the first wave is just starting to enter the Strait at about 1 h. Note that there are large edge waves trapped on the continental shelf that continue to propagate up and down the coast for many additional hours, and lead to additional waves moving into the Strait at later times. Figure 4 shows the same tsunami at 1.5 and 2.5 h , now focusing on the SJdF region. The peak of the first wave reaches both forecast gauges 901 and 911 at about 2:30, and is followed by additional waves. Figure 5 shows the

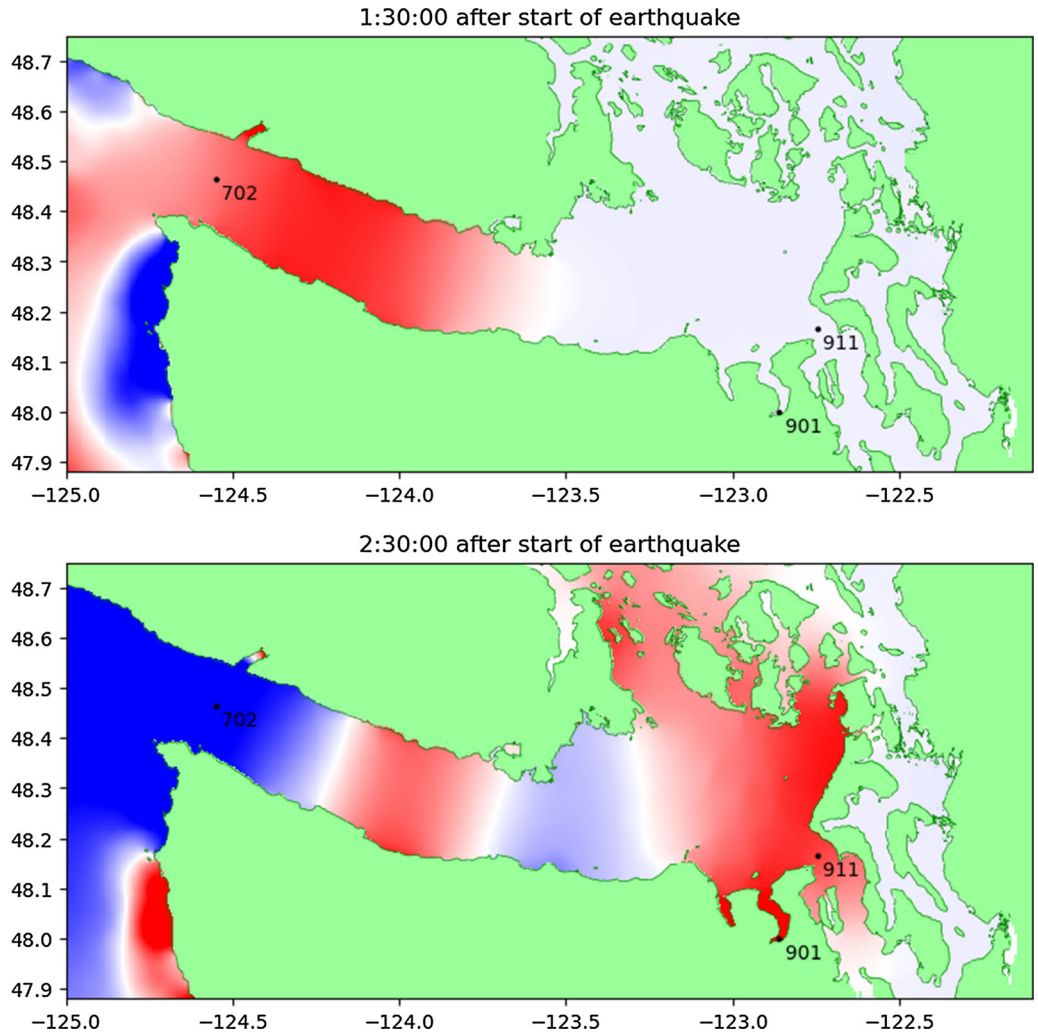

Figure 4
Tsunami from Realization #1127 at times 1:30 and 2:30, with the same colormap as in Fig. 3, shown only over the SJdF region

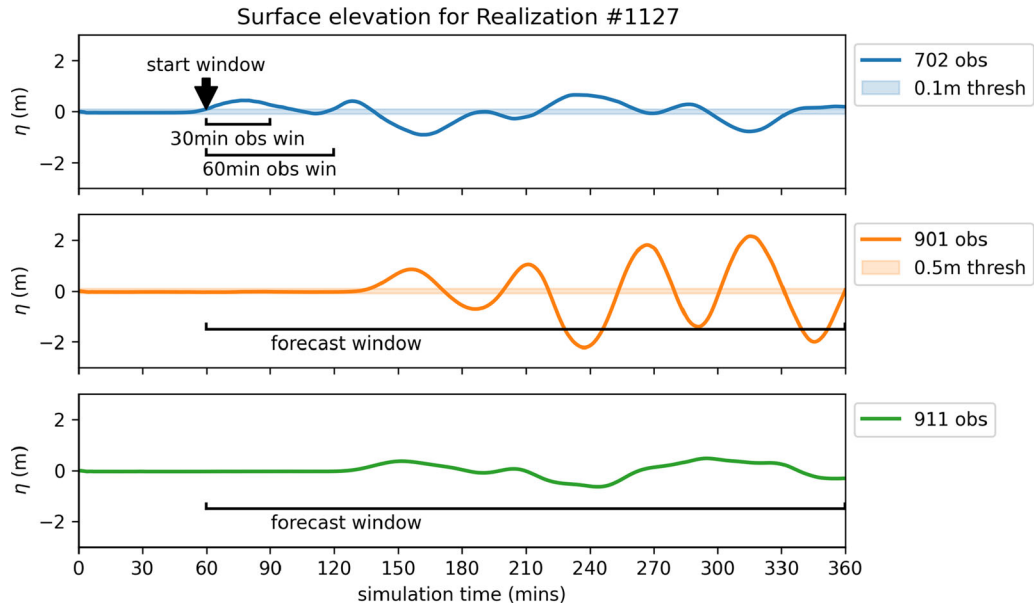

Figure 5
Surface elevation time series measured at the gauges 702, 901, 911 for the Realization #1127, with a depiction of the 30-min and 60-min observation window and the forecast window

time series of surface elevation at these gauges for this realization.

For each of the approaches discussed below, we use some of the random realizations as training data and reserve others for use as test data. There is a great deal of variation in the characteristics of the different tsunamis generated by the KL expansion. However, to minimize concerns that the method trained on KL realizations will only do a good job of predicting other such realizations, we have also tested each model on some additional hypothetical tsunamis generated from seafloor deformations developed by other research groups using completely different approaches, and for which only the final static seafloor deformation is available rather than a time-dependent rupture. In particular, we use a source that models the last major CSZ event of January 26, 1700 that was developed by Wang et al. (2013) to be consistent with the paleotsunami data available from fieldwork along the coast. This model also used a slightly different fault geometry than the one used in generating the training data. Two whole-length CSZ source models from the work of Gao et al. (2018) have also been used, those denoted by B-Whole (a buried rupture) and S-A-Whole (with a splay fault that enhances seafloor uplift). We also use the so-called L1 source from a set of tsunami sources developed for PTHA by Witter et al. (2013), employing a version that was extended further to the north for recent hazard assessment on the Washington coast, as used for example by LeVeque et al. (2020). This is a larger magnitude event that also includes a splay fault. All of these test sources specify instantaneous uplift rather than a propagating rupture and are quite different from the events used as training data for the ML algorithms, particularly those that involve splay faults. In spite of this we find that the algorithms we present generally give surprisingly good forecasts even for these events.

3. Methodology

We first discuss the structure of the data and how we pre-process the data set before applying the various machine learning methods. We will also define the general cost function framework that we will use

to describe various ML models in the subsequent sections.

3.1. Data Preparation

From the 1300 synthetic events of (Melgar, 2016), we first discarded all realizations for which the resulting tsunami was considered negligible in the region of interest. We exclude from our experiments any realization for which the tsunami surface elevation at gauge 702 does not exceed 0.1m in amplitude, or where the amplitude at gauge 901 does not exceed 0.5m during the entirety of simulation time. This reduced the number of realizations from 1300 to 959.

The time series at each gauge that is output by the GeoClaw tsunami simulation may be at non-uniform times, due to the adaptive mesh refinement algorithms used, so they were interpolated to a uniform

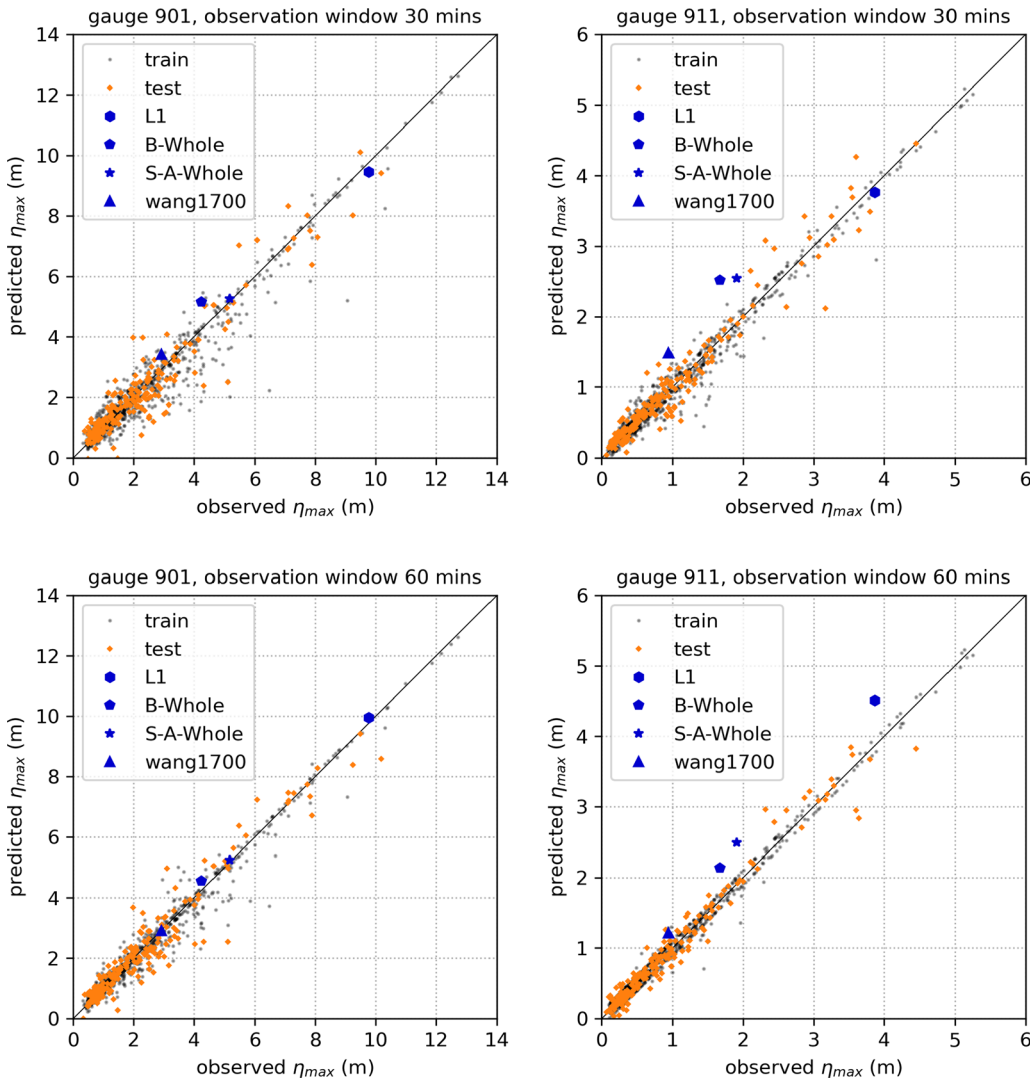

Figure 6
Prediction results using the SVM at gauge 901 (left column) and 911 (right column). Prediction results are compared when using an observation window $T_{ow} = 30$ min of time series data (top row) and 60 min of time series data (bottom row). In each case the scatter plot shows the predicted maximum tsunami elevation vs. the observed maximum from the full simulation, with one point for each realization tested

set of times with 10 second resolution in order to make equal size data sets from each simulation (comparable to the sampling rate of some instruments).

As the observed data we want to use only a short time series of duration T_{ow} (the *observation window*) in order to predict the time series over a longer time period T_{fw} (the *forecast window*) at each forecast gauge. In this paper we compare observation windows of 30 and 60 min and use a prediction window of 5 h. The starting time of these windows is allowed to vary by realization, however, since the tsunami from an earthquake that is more distant from the Strait takes more time to arrive (both at the observation gauge and at the forecast points). We choose the starting time t_1 for each event as the time when the surface amplitude at Gauge 702 first exceeds 0.1m in magnitude. The time series data at gauge 702 from t_1 to $t_1 + T_{ow}$ is then used as the observed data for this event, and we attempt to forecast over time t_1 to $t_1 + T_{fw}$ at each forecast gauge.

From our 959 realizations we randomly chose a training dataset of 767 realizations (80%), and reserved 192 (20%) for testing (along with the additional test sources mentioned in Sect. 2). We use the same splitting of training/test data for each of the algorithms we tested, so that the different ML methods we considered were all presented with the same set of training data. (We also tried different random choices of the training set and obtained similar results.) Note that Figs. 6, 9, and 18 below all show scatter plots of predicted vs. observed maximum surface elevation for each sample event, from both the training set and the test set. These figures show that the test set is well distributed to cover both small and large events within the range of events covered by the training data.

3.2. General Learning Framework

We will set up a general learning framework by introducing definitions and notations that will be used throughout the following sections.

The input and output data are composed of the surface elevation time series in the observation and forecast windows (Figure 5). The input $\mathbf{x} \in \mathbb{R}^{N_{\text{obs}}}$ contains the surface elevation time series of the observation window at gauge 702 at N_{obs} uniform times. The output $\mathbf{y} \in \mathbb{R}^{N_{\text{prd}} \times N_{\text{tg}}}$ contains information about surface elevation in the forecast window at N_{tg} target gauges; it can be either the maximum surface elevation over the window or the time series of surface elevation at N_{prd} uniform times.

As discussed in the previous section, our training dataset contains $N_{\text{train}} = 767$ realizations and it can be written as the pair $\mathcal{D}_{\text{train}} = [\mathbf{X}_{\text{train}}, \mathbf{Y}_{\text{train}}]$ where

$$\mathbf{X}_{\text{train}} = [\mathbf{x}_1, \ldots, \mathbf{x}_{N_{\text{train}}}], \ \mathbf{Y}_{\text{train}} = [\mathbf{y}_1, \ldots, \mathbf{y}_{N_{\text{train}}}]. \tag{1}$$

The test dataset $\mathcal{D}_{\text{test}} = [\mathbf{X}_{\text{test}}, \mathbf{Y}_{\text{test}}]$ of size $N_{\text{test}} = 192$ is defined similarly. The entire data set of size $N_{\text{data}} = N_{\text{train}} + N_{\text{test}}$ will be denoted by $\mathcal{D}_{\text{data}} = [\mathbf{X}_{\text{data}}, \mathbf{Y}_{\text{data}}]$

Our forecast models will be functions of the form

$$f : \mathbb{R}^{N_{\text{obs}}} \times \mathbb{R}^{N_{\text{par}}} \to \mathbb{R}^{N_{\text{prd}} \times N_{\text{tg}}}. \tag{2}$$

For each model parameter $\boldsymbol{\theta} \in \mathbb{R}^{N_{\text{par}}}$ the function $f(\cdot, \boldsymbol{\theta})$ maps the input $\mathbf{x}$ to the output $\mathbf{y}$. We will denote by $\mathcal{F}$ the collection of functions of the form (2). Depending on the specific choice of f, the model can utilize various latent (or hidden) variables $\mathbf{z} \in \mathbb{R}^{N_{\text{lat}}}$ that are not explicitly exposed as inputs or outputs of the model f.

For our prediction task, we need to estimate the parameters $\boldsymbol{\theta}$ by a supervised learning procedure. We define the loss function $\mathcal{L} : \mathcal{F} \times \mathbb{R}^{N_{\text{par}}} \times \mathbb{R}^{N_{\text{obs}}} \times \mathbb{R}^{N_{\text{out}}} \to \mathbb{R}_+$ as a function of the model, the parameters, the training input and the training output. Our loss function $\mathcal{L}$ will be the sum of the data misfit term $\mathcal{L}^c$ and the regularization term $\mathcal{L}^r$, that is,

$$\mathcal{L}(f, \boldsymbol{\theta}, \mathbf{X}, \mathbf{Y}) = \mathcal{L}^c(f, \boldsymbol{\theta}, \mathbf{X}, \mathbf{Y}) + \mathcal{L}^r(f, \boldsymbol{\theta}, \mathbf{X}, \mathbf{Y}). \tag{3}$$

Given a class of regression models f and the training dataset $\mathcal{D}_{\text{train}}$ one learns the parameters $\boldsymbol{\theta}$ by minimizing the the general objective

$$\min_{\boldsymbol{\theta} \in \mathbb{R}^{N_{\text{par}}}} \mathcal{L}(f, \boldsymbol{\theta}, \mathbf{X}_{\text{train}}, \mathbf{Y}_{\text{train}}). \tag{4}$$

In the subsequent sections, we will describe various learning procedures by specifying the model f and the cost functions $\mathcal{L}^c$ and $\mathcal{L}^r$ that appear in this general formulation.

4. Support Vector Machines

Support vector machines (SVMs) are supervised learning models that can be used to perform nonlinear regression and classification on high dimensional data. In our approach, we use ε-Support Vector Regression (ε-SVR) as implemented in the `scikit-learn` library (Pedregosa et al., 2011) to map the input time series to the maximum tsunami amplitude at each forecast location.

For each of the two forecast gauges, we construct two models corresponding to the length of the observation window. For each model, the form of the input is $\mathbf{x} \in \mathbb{R}^{N_{\text{obs}}}$, the time series of the observation window of length 30 or 60 min ($N_{\text{obs}} = 180$ or $N_{\text{obs}} = 360$, respectively), and the output $\mathbf{y} \in \mathbb{R}^{N_{\text{tg}}}$ is the maximum surface elevation $\eta_{\max}$ ($N_{\text{prd}} = 1$) at either of the gauges gauges 901 or 911 ($N_{\text{tg}} = 1$) over the forecast window.

We found that it is more effective to first map each input series into a feature space determined by statistics and other characteristics of the time series. We use the `tsfresh` library to extract features from the time series, such as its distribution of points, correlation properties, stationarity, entropy, and other nonlinear characteristics (Christ et al., 2016, 2018). A vector containing the extracted features $\mathbf{z}$ obtained by applying time characterization methods ψ_j, $j = 1, \ldots, N_{\text{feat}}$ to the time series $\mathbf{x}$ will be denoted by

$$\mathbf{z} = \boldsymbol{\Psi}(\mathbf{x}) = [\psi_1(\mathbf{x}), \ldots, \psi_{N_{\text{feat}}}(\mathbf{x})]. \tag{5}$$

We will collect these features for the training set

$$\mathbf{Z}_{\text{train}} = [\mathbf{z}_1, \ldots, \mathbf{z}_{N_{\text{train}}}] = [\boldsymbol{\Psi}(\mathbf{x}_1), \ldots, \boldsymbol{\Psi}(\mathbf{x}_{N_{\text{train}}})]. \tag{6}$$

The ε-SVR fits a hyperplane of the form $\boldsymbol{\theta}_1^T \mathbf{z} + \boldsymbol{\theta}_2 = 0$ which maximizes the number of training points in the data set $[\mathbf{Z}_{\text{train}}, \mathbf{Y}_{\text{train}}]$ located at most ε distance away from the hyperplane. Collecting the parameters in $\boldsymbol{\theta} = [\boldsymbol{\theta}_1, \boldsymbol{\theta}_2]$, our SVM model is of the form

$$f_{\text{S}} \in \mathcal{F}, \quad f_{\text{S}}(\mathbf{x}; \boldsymbol{\theta}) = \boldsymbol{\theta}_1^T (\boldsymbol{\Phi} \circ \boldsymbol{\Psi})(\mathbf{x}) + \boldsymbol{\theta}_2. \tag{7}$$

The primal problem for estimating model parameters $\boldsymbol{\theta}$ has the form (4) with the hyper-parameters $\varepsilon, C > 0$ (Hastie et al., 2001),

$$\begin{aligned} &\mathcal{L}^c(f_{\text{S}}, \boldsymbol{\theta}, \mathbf{X}_{\text{train}}, \mathbf{Y}_{\text{train}}) \\ &\quad = C \cdot \frac{1}{N_{\text{train}}} \sum_{i=1}^{N_{\text{train}}} \max(0, |f_{\text{S}}(\mathbf{x}_i; \boldsymbol{\theta}) - \mathbf{y}_i| - \varepsilon), \end{aligned} \tag{8}$$

$$\mathcal{L}^r(f_{\text{S}}, \boldsymbol{\theta}, \mathbf{X}_{\text{train}}, \mathbf{Y}_{\text{train}}) = \frac{1}{2} \|\boldsymbol{\theta}\|_{\ell^2}^2.$$

The regularization parameter C controls how strongly training points greater than ε distance from the hyperplane are penalized. The transformation $\boldsymbol{\Phi}(\mathbf{z})$ is in general nonlinear and implicitly maps the feature vectors to a higher dimensional space and gives the algorithm the flexibility to perform nonlinear regression. The resulting optimization problem is solved by applying an SMO-type decomposition method to the associated dual problem (Chen et al., 2006). Rather than specifying ϕ, we apply the kernel-trick and solve the dual problem that only requires $\boldsymbol{\Phi}(\mathbf{z}_i) \cdot \boldsymbol{\Phi}(\mathbf{z}_j)$ which can be represented by the kernel function $K(\mathbf{z}_i, \mathbf{z}_j)$ (Hastie et al., 2001). For our model, we chose the radial basis function (RBF) kernel $K : \mathbb{R}^{N_{\text{feat}}} \times \mathbb{R}^{N_{\text{feat}}} \longrightarrow \mathbb{R}_+$

$$K(\mathbf{z}, \mathbf{z}') = \exp(-\gamma \|\mathbf{z} - \mathbf{z}'\|^2). \tag{9}$$

with regularization parameter $\gamma > 0$.

A grid-based parameter search using k-fold cross-validation (Hastie et al., 2001) was done on the hyper-parameters C and γ in order to obtain the best possible model. The elapsed real time for extracting features and training an SVM was about 6 min, on Intel Xeon Platinum 8268 24C 205W 2.9 GHz Processor and 384 GB of memory. We trained four such models, one for each of the 30 min and 60 min observation windows, predicting $\eta_{\max}$ at gauges 901 and 911. The number of features used was $N_{\text{feat}} = 700$. The results are shown in Fig. 6.

We have also tried a variation of this approach, by inputting $\mathbf{x}$ directly in an SVM, in place of the extracted features $\boldsymbol{\Psi}(\mathbf{x})$. We also experimented with a random forest, another widely used regression model (Hastie et al., 2001). A summary comparing these approaches appear in Sect. 7.

5. Denoising Autoencoder (DAE)

As another approach, we will use a type of neural network called the *denoising autoencoder* to forecast the full time series of the surface elevation in the forecast window, as opposed to just the maximum surface elevation over the window. We will denote the model by f_{D} in the form (2). The input $\mathbf{x} \in \mathbb{R}^{N_{\mathrm{obs}}}$ contains the interpolated surface elevation time series of gauge 702 in the observation window at uniform times, of sizes $N_{\mathrm{obs}} = 26$ and $N_{\mathrm{obs}} = 51$, for the 30 min and 60 min windows, respectively. The model output $\mathbf{y} \in \mathbb{R}^{N_{\mathrm{prd}} \times N_{\mathrm{tg}}}$ will be the estimate for the full time series for the surface elevation at all gauges ($N_{\mathrm{tg}} = 3$) over the 5 h forecast window at uniform times ($N_{\mathrm{prd}} = 256$). We will use the same time-discretizations for $\mathbf{x}$ and $\mathbf{y}$; as a result $\mathbf{x}_i$ will be a sub-block of $\mathbf{y}_i$ for each realization.

A DAE is a neural network with the encoder-decoder structure that is trained to denoise or correct corrupted data (Goodfellow et al., 2016). We will use h and g to denote two neural networks, called the encoder (or recognition model) and the decoder (or generative model) respectively. The input and output to the DAE will be distinct from those of our model f_D, so we will use $\mathbf{x}'$ to denote the input to the DAE to distinguish it from the input $\mathbf{x}$ to the model f_D.

The encoder $h : \mathbb{R}^{N_{\mathrm{prd}}} \times \mathbb{R}^{N_{\mathrm{par}}^{(h)}} \longrightarrow \mathbb{R}^{N_{\mathrm{lat}}}$ maps the DAE input $\mathbf{x}' \in \mathbb{R}^{N_{\mathrm{prd}}}$ to latent variables $\mathbf{z} \in \mathbb{R}^{N_{\mathrm{lat}}}$ while the decoder $g : \mathbb{R}^{N_{\mathrm{lat}}} \times \mathbb{R}^{N_{\mathrm{par}}^{(g)}} \longrightarrow \mathbb{R}^{N_{\mathrm{prd}}}$ maps the latent variables to a reconstruction of the DAE input. Writing $\boldsymbol{\theta}_h$ to denote parameters of h and $\boldsymbol{\theta}_g$ that of g, we let $\boldsymbol{\theta} = [\boldsymbol{\theta}_h, \boldsymbol{\theta}_g]$ and $N_{\mathrm{par}} = N_{\mathrm{par}}^{(h)} + N_{\mathrm{par}}^{(g)}$. An *autoencoder* is the neural network resulting from the composition

$$\begin{aligned} &\chi_D : \mathbb{R}^{N_{\mathrm{prd}}} \times \mathbb{R}^{N_{\mathrm{par}}} \longrightarrow \mathbb{R}^{N_{\mathrm{prd}}}, \\ &\chi_D(\mathbf{x}'; \boldsymbol{\theta}) = g(h(\mathbf{x}'; \boldsymbol{\theta}_h); \boldsymbol{\theta}_g). \end{aligned} \tag{10}$$

Autoencoders can be trained to denoise its input data, simply by corrupting the training input $\mathbf{X}'_{\mathrm{train}}$ during the learning procedure. Let $\widetilde{\mathbf{X}}'_{\mathrm{train}}$ denote a corrupted version of the training input data $\mathbf{X}'_{\mathrm{train}}$ then the autoencoder χ_D can be trained to recover the uncorrupted data $\mathbf{x}'$ from $\tilde{\mathbf{x}}'$ via the procedure

$$\min_{\boldsymbol{\theta} \in \mathbb{R}^{N_{\mathrm{par}}}} \mathcal{L}(\chi_D, \boldsymbol{\theta}, \widetilde{\mathbf{X}}'_{\mathrm{train}}, \mathbf{X}'_{\mathrm{train}}). \tag{11}$$

The resulting DAE χ_D is a nonlinear denoising function. The ability of DAEs to recover the uncorrupted data goes beyond random noise corruption (Vincent et al., 2010) to more severe or structured forms of corruption; for example, a significant contiguous portion of the input data can be masked (Pathak et al., 2016; Li et al., 2017).

The task of predicting future surface elevation at various gauge locations can be put into this framework. In the context of our forecast model f_D, the desired output $\mathbf{y} \in \mathbb{R}^{N_{\mathrm{prd}} \times N_{\mathrm{tg}}}$ will play the role of the uncorrupted DAE input variable $\mathbf{x}'$. That is, the desired reconstruction from the DAE is the full time series in the forecast window at all gauges, the surface elevation at N_{prd} uniform times. For example, for the i-th data point in the training set $\mathcal{D}_{\mathrm{train}}$ the uncorrupted DAE input is

$$\mathbf{x}'_i = \begin{bmatrix} \boldsymbol{\eta}_{702,i} \\ \boldsymbol{\eta}_{901,i} \\ \boldsymbol{\eta}_{911,i} \end{bmatrix} = \begin{bmatrix} \eta_{702,i,1} & \cdots & \eta_{702,i,n} & \cdots & \eta_{702,i,N_{\mathrm{prd}}} \\ \eta_{901,i,1} & \cdots & \eta_{901,i,n} & \cdots & \eta_{901,i,N_{\mathrm{prd}}} \\ \eta_{911,i,1} & \cdots & \eta_{911,i,n} & \cdots & \eta_{911,i,N_{\mathrm{prd}}} \end{bmatrix}. \tag{12}$$

This is equal to the desired output $\mathbf{y}_i$ of the forecast model f_D.

In our prediction task, we use the surface elevation reading at gauge 702 in the observation window for prediction. This can be viewed as partial data of the time series over the longer forecast window. For the i-th data point, the corresponding corrupted data of has the form

$$\begin{bmatrix} \eta_{702,i,1} & \cdots & \eta_{702,i,t_{N_{\mathrm{pred}}}} & 0 & \cdots & 0 \\ 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & \cdots & 0 & 0 & \cdots & 0 \end{bmatrix}. \tag{13}$$

In other words, we mask the data by setting to zero the entries outside the observation window. In addition, we will choose to omit the null rows, after which the i-th corrupted DAE input $\tilde{\mathbf{x}}'_i$ becomes

$$\tilde{\mathbf{x}}'_i = \begin{bmatrix} \eta_{702,i,1} & \cdots & \eta_{702,i,N_{\mathrm{obs}}} & 0 & \cdots & 0 \end{bmatrix} \in \mathbb{R}^{N_{\mathrm{prd}}}. \tag{14}$$

Relating this to the input $\mathbf{x}_i$ to our forecast model, $\tilde{\mathbf{x}}'_i$ is equal to $[\mathbf{x}_i \mid \mathbf{0}]$. A zero-padding map A can be defined to relate the two in a concise manner,

Table 1

Specification of number of input/output channels for convolution kernels $\mathbf{K}_{\text{enc}}^{(\ell)}, \mathbf{K}_{\text{dec}}^{(\ell)}$ *for layer* ℓ

ℓ	1	2	3	4	5	6	7	8
In channels for $\mathbf{K}_{\text{enc}}^{(\ell)}$	1	64	64	128	128	256	256	512
Out channels $\mathbf{K}_{\text{enc}}^{(\ell)}$	64	64	128	128	256	256	512	512
In channels for $\mathbf{K}_{\text{dec}}^{(\ell)}$	3	64	64	128	128	256	256	512
Out channels $\mathbf{K}_{\text{dec}}^{(\ell)}$	64	64	128	128	256	256	512	512

$$A : \mathbb{R}^{N_{\text{obs}}} \to \mathbb{R}^{N_{\text{prd}}}, \quad A(\mathbf{x}_i) = [\,\mathbf{x}_i \quad | \quad \mathbf{0}\,]. \tag{15}$$

Finally, we define our model f_{D} by composing the zero-padding map A and the DAE χ_D,

$$f_{\text{D}} \in \mathcal{F}, \quad f_{\text{D}}(\mathbf{x};\boldsymbol{\theta}) = \chi_D(A(\mathbf{x});\boldsymbol{\theta}). \tag{16}$$

That is, the model f_D takes the time series data from the observation window and feeds the zero-padded partial data into the DAE, which in turn reconstructs the full time series in the forecast window.

We next specify the architectures for the neural networks h and g. Here we will employ standard convolutional neural networks (CNNs) for both the encoder h and the decoder g network. CNNs form a fundamentally important class of neural networks that enabled breakthrough results in signal and image processing (LeCun 1989). In the discussion here we use standard technical terminology without detailed explanations, see references such as Goodfellow et al. (2016).

The network architecture for the encoder h can be written as follows,

$$h(\tilde{\mathbf{x}}';\boldsymbol{\theta}_h) = P \circ \sigma \circ \mathbf{K}_{\text{enc}}^{(L)}(\boldsymbol{\theta}_h) * \cdots * P \circ \sigma \circ \mathbf{K}_{\text{enc}}^{(1)}(\boldsymbol{\theta}_h) * \tilde{\mathbf{x}}'. \tag{17}$$

Here $*$ denotes the 1D convolution operation, σ the activation function, P the pooling operator.

The decoder g is typically defined as the "transpose" of the encoder,

$$g(\mathbf{z};\boldsymbol{\theta}_g) = \sigma \circ \mathbf{K}_{\text{dec}}^{(L)}(\boldsymbol{\theta}_g) *' \cdots *' \sigma \circ \mathbf{K}_{\text{dec}}^{(1)}(\boldsymbol{\theta}_g) *' \mathbf{z}, \tag{18}$$

where $*'$ denotes the transpose of the convolution operator $*$.

We choose the convolution kernel $\mathbf{K}_{\text{enc}}^{(\ell)}$ of width 3 with bias and zero-padding, and $\mathbf{K}_{\text{dec}}^{(\ell)}$ of width 2 and stride 2 with bias. This choice was made to be consistent with the input/ouput signal size of $N_{\text{prd}} = 256$, but can be adapted to match an arbitrary signal sizes in a straightforward manner. The number of channels for each layer is shown in Table 1. The activation σ is chosen as the leaky rectified linear unit (LReLU) with negative slope 0.5, and the pooling operator P as the max-pool operator with stride 2. The networks h and g respectively have the total number of parameters $N_{\text{par}}^{(h)} = 1.5$ million and $N_{\text{par}}^{(g)} = 1$ million. A simple block-diagram of the DAE architecture is displayed in Fig. 7. This architecture was inspired by a 2D analogue used for image processing tasks (Li et al., 2017). The large number of parameters for the model f_D is unusual from a classical perspective, but is common in deep learning models (Zhang et al., 2017).

We use the mean absolute error (MAE) as the training loss without any regularization,

$$\begin{aligned} &\mathcal{L}^c(f_{\text{D}}, \boldsymbol{\theta}, \mathbf{X}_{\text{train}}, \mathbf{Y}_{\text{train}}) \\ &\quad = \frac{1}{N_{\text{train}}} \sum_{i=1}^{N_{\text{train}}} \|\mathbf{y}_i - f_{\text{D}}(\mathbf{x}_i, \boldsymbol{\theta})\|_{\ell^1}^2, \\ &\mathcal{L}^r = 0. \end{aligned} \tag{19}$$

When training deep neural networks, one typically partitions the training set into minibatches (Goodfellow et al., 2016). We choose minibatches each with at most 20 data points. The estimation of the parameters $\boldsymbol{\theta}$ for the encoder-decoder pair is done by solving the problem (4) via a stochastic gradient descent algorithm *Adam* developed by Kingma and Ba (2015).

We are also interested in estimating the uncertainty in our forecasts, so we train an ensemble of

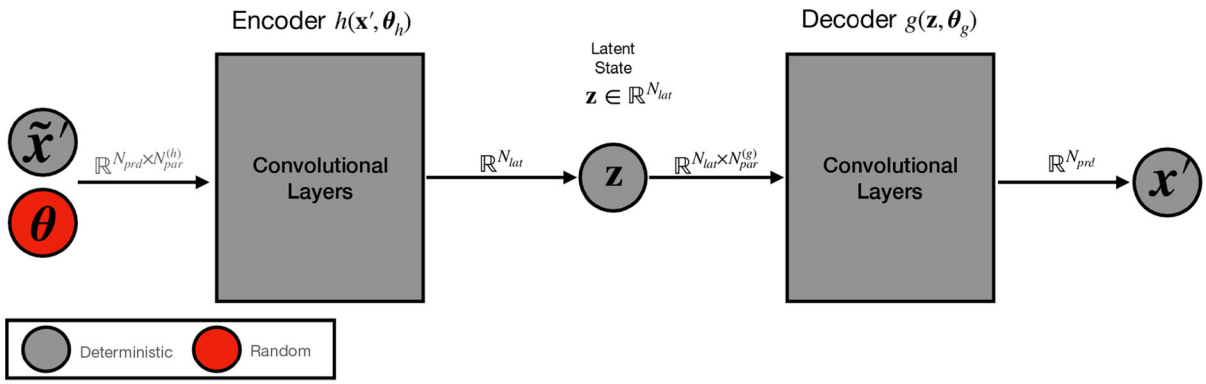

Figure 7
A diagram of the convolutional architecture of the deterministic autoencoder

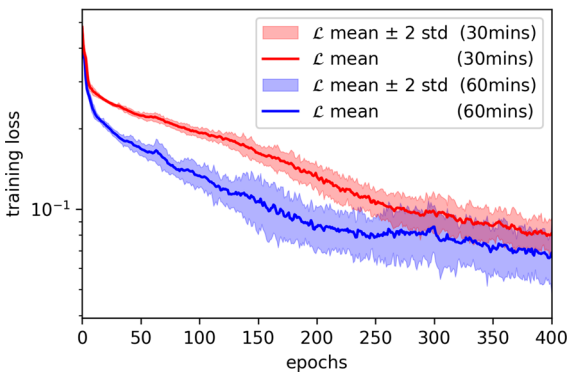

Figure 8
The mean and two standard deviations of ensemble training loss of the ensemble during stochastic gradient descent

autoencoders on an identical training set, each from from randomly initialized set of weights. For example, the ensemble would consist of a population of models $\{f_{\mathrm{D}}^{(k)}\}_{k=1}^{N_{\mathrm{ens}}}$. To make a prediction, we use the mean and variance of the ensemble. For a new input $\mathbf{x}$, we use the sample mean of the autoencoder output, and the uncertainty regarding the prediction is provided by the sample variance

$$\hat{\boldsymbol{\mu}} = \mathbb{E}[f_{\mathrm{D}}(\mathbf{x})], \quad \hat{\boldsymbol{\sigma}}^2 = \mathrm{Var}[f_{\mathrm{D}}(\mathbf{x})]. \tag{20}$$

We will take $\hat{\boldsymbol{\mu}}$ as our prediction $\hat{\mathbf{y}}$. The ensemble output represents the uncertainty in parameter estimation (19), and can be viewed as providing an estimate of the uncertainty in the forecast. However, there is no rigorous guarantee that the observation will lie within certain distance to the estimate $\hat{\mathbf{y}}$.

We train the ensemble up to 400 epochs. The training loss of the ensemble at each epoch is plotted in Fig. 8. The loss tends to decrease more slowly when only 30-min window is used, reaching comparable levels around 250 epochs. Our implementation is based on the `PyTorch` package (Paszke et al., 2019). The elapsed real time for training one model for 400 epochs using Intel Xeon Platinum 8268 24C 205W 2.9 GHz Processor and 384 GB of memory with NVIDIA V100 GPUs is about 3 min. We trained two ensembles of 25 models each ensemble using 30 min and 60 min of observation windows as inputs. The individual models in the ensemble can be trained in parallel, but in this work we trained the models sequentially on a single machine. We used the neural network trained up to 100 and 200 epochs, for the 30 min and 60 min models respectively, to produce the results.

The DAE produces a full time series over the 5 h window as the result of its prediction, so we can also estimate the maximum surface elevation over the time series to predict $\eta_{\max}$ for each gauge. The resulting prediction for the training and test data are displayed in a scatter plot in Fig. 9. The prediction is in good agreement with the observed data, and the prediction generally improves when a larger observation window is used.

The full time series prediction at Gauge 901 for a specific run in the dataset is shown in Fig. 10. This

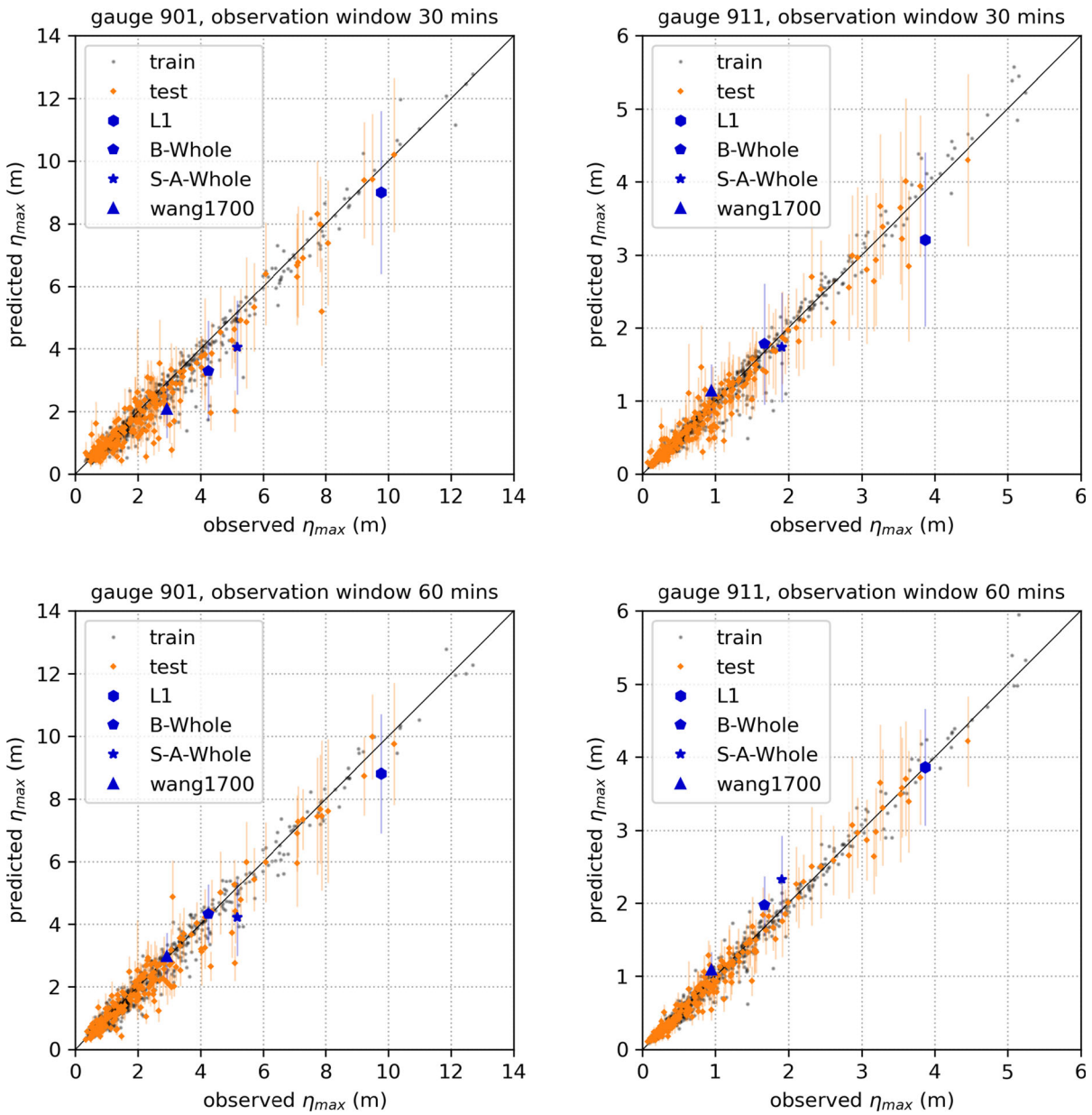

Figure 9
Prediction results using the DAE, as explained in the caption of Fig. 6, with the addition of a vertical bar centered on each test prediction showing 2σ for the uncertainty in the prediction, as explained in Sect. 5

figure shows the 25 predictions from the ensemble, each of which was generated using a DAE model trained with a different initial guess for the parameters, providing an estimate of the uncertainty as described above. The mean prediction, along with 2σ uncertainty bands, are then shown in Fig. 11 for all gauges, and for both observation windows. Predictions for the L1 scenario are shown in Fig. 12. Both show good agreement with the observations. We note that the uncertainty illustrated here shows the

uncertainty within this particular ML model (by estimating the sensitivity to the choice of initial guess of parameters). It does not directly provide an estimate of the accuracy of the forecast, and the true observation often lies outside of the uncertainty band, particularly at later times and for test realizations that exhibit quite different wave characteristics than the training data used to train the model. Nonetheless, this uncertainty gives some useful information about this particular model regarding the sensitivity or stability of the training procedure (19), which is is related to the well-posedness of the parameter estimation task. For a systematic discussion relating neural network ensembles and uncertainty estimation, see Lakshminarayanan et al. (2017). In the next section, we will consider a neural network architecture that provides a more explicit treatment of the uncertainty.

6. *Variational Autoencoder (VAE)*

As touched on in Sect. 5, we may train an ensemble of models to create confidence bounds on the output. However, we can augment the autoencoder framework to produce these confidence bounds while training only one model. A VAE is an autoencoder whose latent *random* variables $\mathbf{z} \in \mathbb{R}^{N_{\text{lat}}}$ are trained to have desirable probability distributions. Similarly to the deep convolutional autoencoder, the VAE can be decomposed into an probabilistic encoder model and a probabilistic decoder model, and the random latent variables $\mathbf{z}$ sit in between the recognition and generative models. This modification to the usual deterministic autoencoder requires a probabilistic interpretation of the cost function described in Sect. 3. The derivation here follows Kingma and Welling (2014).

We will first extend the deterministic autoencoder model (10) to a stochastic one. We assume that the input data $\mathbf{x}'$ to the autoencoder is generated by a random process involving the latent random variable $\mathbf{z}$. This process first generates a value $\mathbf{z}_i$ from some prior distribution $p(\mathbf{z})$ then generates $\mathbf{x}'_i$ from a conditional distribution $p(\mathbf{x}'|\mathbf{z})$.

We will assume that the prior $p(\mathbf{z})$ and likelihood $p(\mathbf{x}'|\mathbf{z})$ come from parametric families of distributions $p(\mathbf{z};\boldsymbol{\vartheta})$ and $p(\mathbf{x}'|\mathbf{z};\boldsymbol{\vartheta})$. We do not know the true parameters $\boldsymbol{\vartheta}^*$ nor $\mathbf{z}$, so we perform a maximum likelihood or maximum a posteriori (MAP) inference on the global parameters and variational inference on the latent parameters.

Many variational inference models concern evaluating the marginal likelihood $p(\mathbf{x}';\boldsymbol{\vartheta})$ but VAEs also learn the true posterior density

$$p(\mathbf{z}|\mathbf{x}';\boldsymbol{\vartheta}) = \frac{p(\mathbf{x}'|\mathbf{z};\boldsymbol{\vartheta})p(\mathbf{z};\boldsymbol{\vartheta})}{p(\mathbf{x}';\boldsymbol{\vartheta})}. \tag{21}$$

These functions are extremely intractable for nonlinear problems with large dimensions, so the VAEs make approximations: an encoder model $\mathbf{z} \sim q(\mathbf{z}|\mathbf{x};\boldsymbol{\theta}_h)$ is introduced, as an approximation to the intractable true posterior $p(\mathbf{z}|\mathbf{x}';\boldsymbol{\vartheta})$. However, one can learn the encoder model parameters jointly with the decoder model parameters $\boldsymbol{\vartheta}$.

The marginal likelihood is composed of the sum over the marginal likelihoods of individual data points

$$\log p(\mathbf{x}';\boldsymbol{\vartheta}) \approx \sum_{i=1}^{N_{\text{data}}} \log p(\mathbf{x}'_i;\boldsymbol{\vartheta}).$$

Each ith component of the marginal likelihood can be rewritten as

$$\begin{aligned}\log p(\mathbf{x}'_i;\boldsymbol{\vartheta}) &= D_{KL}(q(\mathbf{z}|\mathbf{x}_i;\boldsymbol{\theta}_h) \,\|\, p(\mathbf{z}|\mathbf{x}_i;\boldsymbol{\vartheta})) \\ &\quad - D_{KL}(q(\mathbf{z}|\mathbf{x}_i;\boldsymbol{\theta}_h) \,\|\, p(\mathbf{z};\boldsymbol{\vartheta})) \\ &\quad + \mathbb{E}_{q(\mathbf{z}|\mathbf{x}_i;\boldsymbol{\theta}_h)}[\log p(\mathbf{x}_i|\mathbf{z};\boldsymbol{\vartheta})]\end{aligned} \tag{22}$$

where the first two terms on the RHS are the Kullback-Leibler (KL) divergence given by, e.g.

$$\begin{aligned}&D_{\text{KL}}\left(q(\mathbf{z}|\mathbf{x}'_i;\boldsymbol{\theta}_h) \,\|\, p(\mathbf{z};\boldsymbol{\vartheta})\right) \\ &= \mathbb{E}_{q(\mathbf{z}|\mathbf{x}'_i;\boldsymbol{\theta}_h)}\left[\log \frac{q(\mathbf{z}|\mathbf{x}'_i;\boldsymbol{\theta}_h)}{p(\mathbf{z};\boldsymbol{\vartheta})}\right].\end{aligned} \tag{23}$$

This second term can be interpreted as a regularizer encouraging the approximate posterior to be close to the prior $p(\mathbf{z};\boldsymbol{\vartheta})$. The KL divergence is non-negative, so omitting the first term leads to the variational lower bound on the marginal likelihood of i-th data point,

$$\begin{aligned}\log p(\mathbf{x}'_i;\boldsymbol{\vartheta}) \geq\ &- D_{\text{KL}}\left(q(\mathbf{z}|\mathbf{x}'_i;\boldsymbol{\theta}_h) \,\|\, p(\mathbf{z};\boldsymbol{\vartheta})\right) \\ &+ \mathbb{E}_{q(\mathbf{z}|\mathbf{x}'_i;\boldsymbol{\theta}_h)}\left[\log p(\mathbf{x}'_i|\mathbf{z};\boldsymbol{\vartheta})\right]\end{aligned} \tag{24}$$

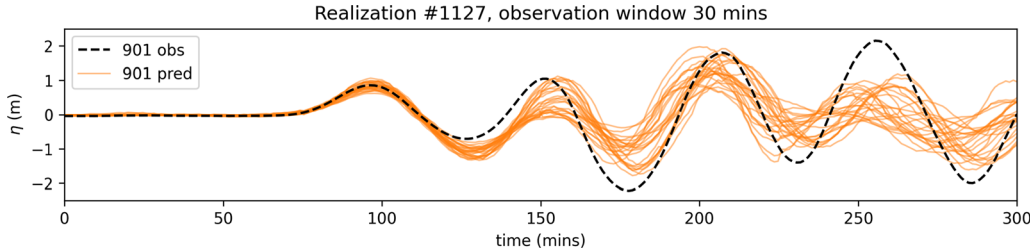

Figure 10
Time series predictions at Gauge 901 from the 25 individual DAE models in the ensemble for Realization #1127, obtained with different initial guesses for the parameters. The mean and two standard deviations are shown in Fig. 11

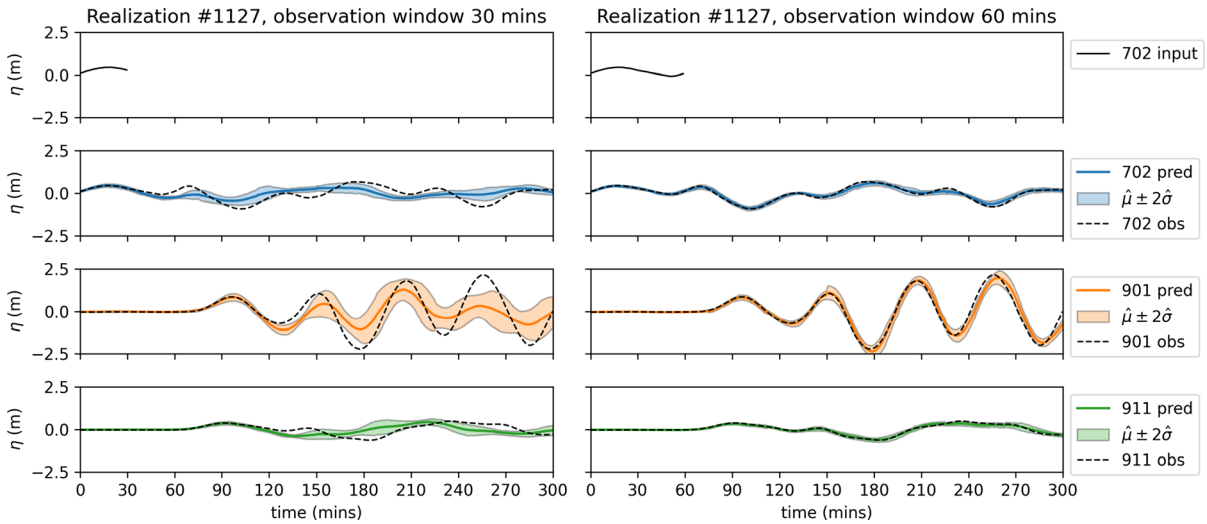

Figure 11
Predicted time series for realization #1127 using the DAE, based on the observed data shown in the top row for observation window 30 min (left column) or 60 min (right column). The sensitivity of the model to the initial guess for the parameters is indicated by the uncertainty bands, as described in the text

This lower bound will serve as our loss function $\mathcal{L}$ in the minimization problem (4).

In this view of the VAE, the encoder h of the form (10) is now a *parametric inference model* $q(\mathbf{z}|\mathbf{x};\boldsymbol{\theta}_h)$, optimized so that $q(\mathbf{z}|\mathbf{x}';\boldsymbol{\theta}_h) \approx p(\mathbf{z}|\mathbf{x}';\boldsymbol{\vartheta})$.

We note here that unbiased gradients are difficult to attain in a straightforward way, so we use an optimization routine derived by the reparameterization trick (Rezende et al., 2014) which expresses $\mathbf{z} \sim q(\mathbf{z}|\mathbf{x};\boldsymbol{\theta}_h)$ as a differentiable and invertible deterministic transformation of another random variable $\boldsymbol{\varepsilon}$ given $\boldsymbol{\theta}_h$ and $\mathbf{x}'$,

$$\mathrm{z} = \mathcal{G}(\boldsymbol{\varepsilon}, \mathbf{x}'; \boldsymbol{\theta}_h) \tag{25}$$

for $\boldsymbol{\varepsilon}$ independent of $\boldsymbol{\vartheta}$ and $\mathbf{x}'$ with its own independent marginal $p(\boldsymbol{\varepsilon})$. This allows us to replace the expectation with respect to $q(\mathbf{z}|\mathbf{x};\boldsymbol{\theta}_h)$ with the expectation with respect to $p(\boldsymbol{\varepsilon})$, thereby rewriting the expectation in the RHS of (24)

$$\begin{aligned}&\mathbb{E}_{q(\mathbf{z}|\mathbf{x};\boldsymbol{\theta}_h)}\left[\log p(\mathbf{x}_i'|\mathbf{z};\boldsymbol{\vartheta})\right]\\&= \mathbb{E}_{p(\boldsymbol{\varepsilon})}\left[\log p(\mathrm{z};\boldsymbol{\vartheta}) p(\mathbf{x}_i'|\mathrm{z};\boldsymbol{\vartheta}) - \log q(\mathrm{z}|\mathbf{x}_i';\boldsymbol{\theta}_h)\right]\end{aligned} \tag{26}$$

where $\mathrm{z} = \mathcal{G}(\boldsymbol{\varepsilon}, \mathbf{x}')$. This allows the formation of a Monte Carlo estimator of the individual data point

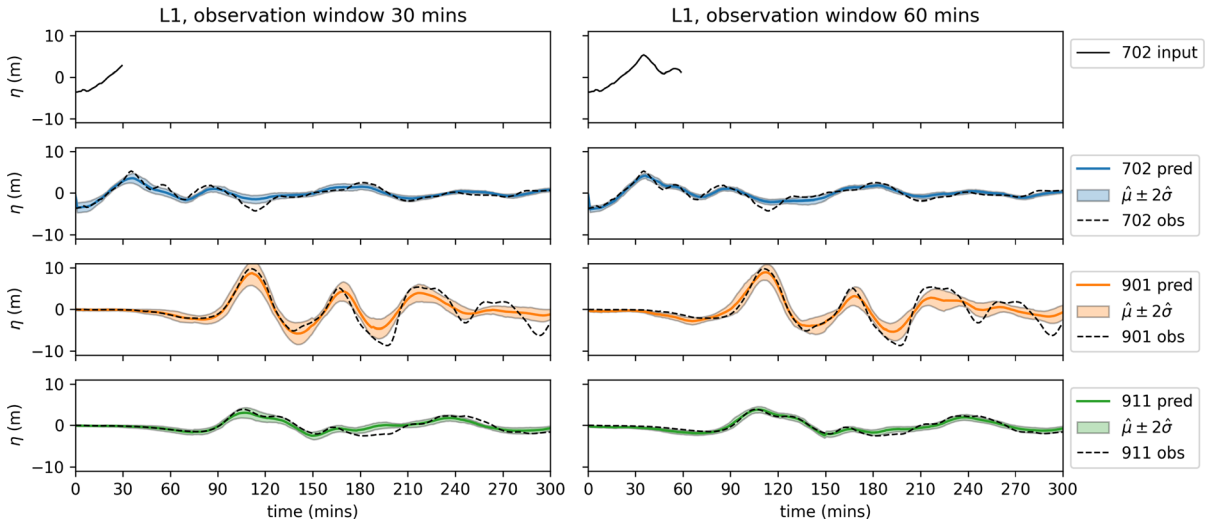

Figure 12
Predicted time series for the L1 scenario using the DAE, based on the observed data shown in the top row for observation window 30 min (left column) or 60 min (right column)

evidence lower bound where we use a single noise sample $\boldsymbol{\varepsilon}$ from $p(\boldsymbol{\varepsilon})$ that is unbiased.

A common prior that we choose here is the centered isotropic Gaussian $p(\mathbf{z};\boldsymbol{\vartheta}) = \mathcal{N}(\mathbf{z};\mathbf{0},\mathbf{I})$ where $\mathbf{I}$ is the identity matrix. We let the variational approximate posterior $q(\mathbf{z}|\mathbf{x};\boldsymbol{\theta}_h)$ be the multivariate Gaussian with diagonal covariance $q(\mathbf{z}|\mathbf{x}_i;\boldsymbol{\theta}_h) = \mathcal{N}(\mathbf{z};\boldsymbol{\mu}_i,\boldsymbol{\sigma}_i^2)$ with $\mathbf{z} \in \mathbb{R}^{N_{\text{lat}}}$ for $N_{\text{lat}} = 450$, and the mean and standard deviation $\boldsymbol{\mu}_i$ and $\boldsymbol{\sigma}_i$ are outputs of the encoder $h(\mathbf{x}';\boldsymbol{\theta}_h)$.

Finally, we use the zero-padding function A (15) with input $\mathbf{x}_i \in \mathbb{R}^{N_{\text{prd}} \times N_{\text{tg}}}$ and output $\mathbf{y}_i \in \mathbb{R}^{N_{\text{prd}} \times N_{\text{tg}}}$ as in the previous section. We then sample from the posterior $\mathbf{z}_i \sim q(\mathbf{z}|\mathbf{x}_i';\boldsymbol{\theta}_h)$ as follows,

$$\begin{aligned} \mathbf{x}_i' &= A(\mathbf{x}_i) \\ (\boldsymbol{\mu}_i, \boldsymbol{\sigma}_i) &= h(\mathbf{x}_i';\boldsymbol{\theta}_h), \\ \mathrm{z}_i &= \mathcal{G}(\boldsymbol{\varepsilon}, \mathbf{x}_i';\boldsymbol{\theta}_h) = \boldsymbol{\mu}_i + \boldsymbol{\sigma}_i \odot \boldsymbol{\varepsilon}, \quad \boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0},\mathbf{I}), \\ \hat{\mathbf{y}}_i &= g(\mathrm{z}_i;\boldsymbol{\theta}_g), \end{aligned} \tag{27}$$

where $\odot$ is the Hadamard product. This random process will serve as our model, which we denote by f_V. While the input-output dimensions of f_V are same as the functions in $\mathcal{F}$, we will presume that the output of f_V is random.

The KL divergence has the form

$$\begin{aligned} &-D_{\text{KL}}(q(\mathbf{z}|\mathbf{x}_i;\boldsymbol{\theta}_h) \,\|\, p(\mathbf{z};\boldsymbol{\vartheta})) \\ &\quad = \frac{1}{2}\sum_{j=1}^{N_{\text{lat}}}(1 + \log(\sigma_{j,i}^2) - \mu_{j,i}^2 - \sigma_{j,i}^2). \end{aligned} \tag{28}$$

Let us collect all parameters in $\boldsymbol{\theta} = [\,\boldsymbol{\theta}_h \;\mid\; \boldsymbol{\theta}_g\,]$. Upon rewriting the second term in (24) as an ℓ^2 misfit, we arrive at a problem to that of the form (8),

$$\begin{aligned} &\mathcal{L}^c(f_V, \boldsymbol{\theta}, \mathbf{X}_{\text{train}}, \mathbf{Y}_{\text{train}}) \\ &\quad = \frac{1}{N_{\text{train}}}\sum_{i=1}^{N_{\text{train}}} \|\mathbf{y}_i - f_V(\mathbf{x}_i;\boldsymbol{\theta})\|_{\ell^2}^2, \\ &\mathcal{L}^r(f_V, \boldsymbol{\theta}, \mathbf{X}_{\text{train}}, \mathbf{Y}_{\text{train}}) \\ &\quad = \frac{1}{2}\sum_{i=1}^{N_{\text{train}}}\sum_{j=1}^{N_{\text{lat}}}(1 + \log(\sigma_{j,i}^2) - \mu_{j,i}^2 - \sigma_{j,i}^2). \end{aligned} \tag{29}$$

We train one VAE on the same training set described in Sect. 3. The elapsed real time for training the VAE for 1000 epochs is about 3 min using the same hardware as the DAE. With only one model to train, we were also able to train the VAE on a laptop using a CPU, a 2017 15in MacBook Pro with Intel Core i7 Quad-Core 3.1 GHz Processor with 16 GB memory, the elapsed real time was about 25 min. The

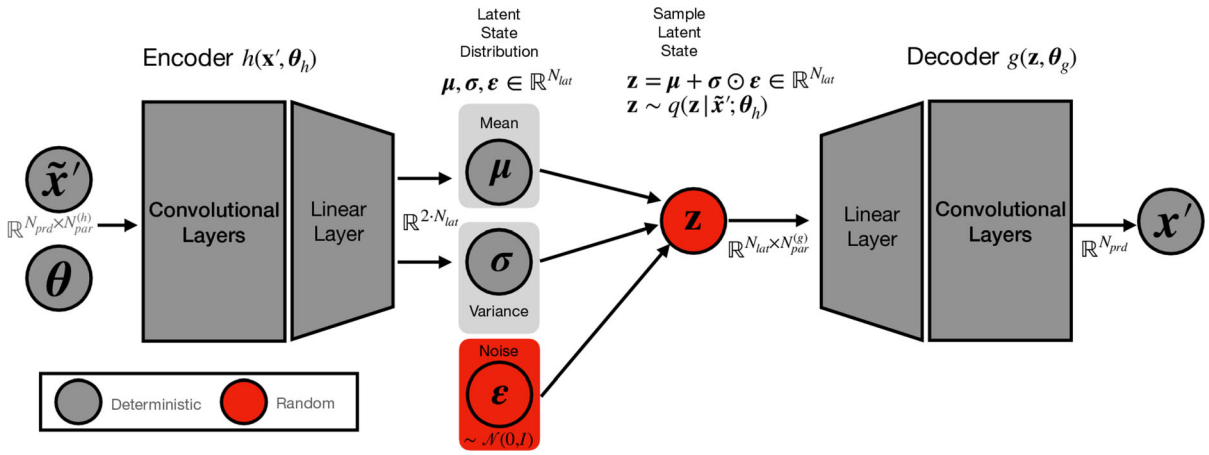

Figure 13
A diagram of the convolutional architecture of the variational autoencoder (27) as described in Sect. 6

training loss for each epoch is plotted in Fig. 14. We trained two VAEs, each providing predictions based on 30 and 60 min observation windows.

The trained model also produces $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$, the mean and standard deviation of the latent variables, for each input. We then create 100 realizations by sampling the latent variables from this Gaussian distribution $\boldsymbol{\varepsilon} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma})$ and evaluate the decoder model $g(\mathrm{z}; \boldsymbol{\theta}_g)$ for each $\mathrm{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\varepsilon}$. This gives us 100 time series, as shown in Fig. 15. From these we can compute the mean time series and a standard deviation at each time, similar to the approach used for the DAE, in order to obtain the forecast gauge results with uncertainty bands shown in Fig. 16 for #1127 and in Fig. 17 for the L1 scenario. As discussed previously for the DAE, the uncertainty bands shown are related to uncertainty in the model being fit to the data, and should not be interpreted as estimates of the error in the forecast. An advantage of the VAE approach is that only one ML model is trained, which results in $(\boldsymbol{\mu}, \boldsymbol{\sigma})$ as an output. It is then very quick to evaluate 100 (or more) realizations in order to compute the mean prediction forecast. By contrast, the mean prediction (and uncertainties) for the DAE was determined by training 25 randomized models, which requires more computational resources. In either case applying the model to new set of observation data is extremely quick and could be done in real time during an event, provided the training was all done in advance.

We can also record the maximum surface elevation from the predicted time series resulting from the VAE for each test realization, and we compare these to the true (observed) maximum surface elevation in the scatter plot of Fig. 18. As with the previous methods tested, we see that prediction is in good agreement with the observed data, and the prediction improves when a larger window is used.

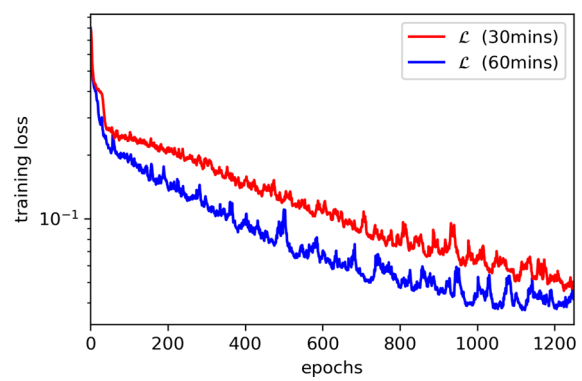

Figure 14
Training loss for each epoch during the training procedure of the VAE

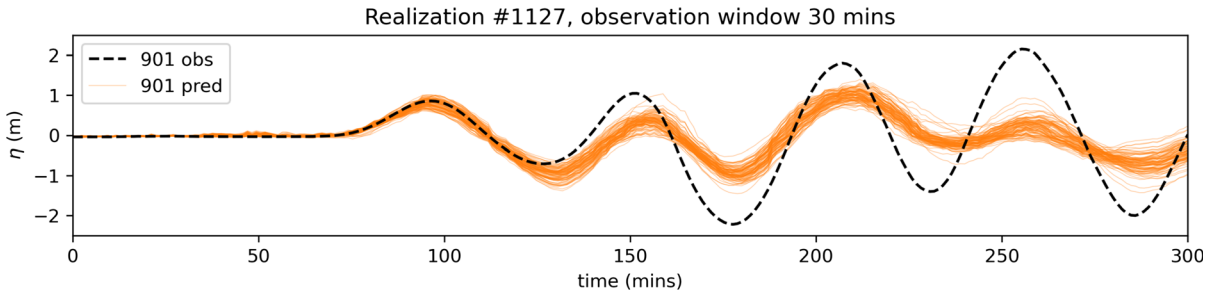

Figure 15
Randomly generated time series predictions from the VAE models for Realization #1127. Shown here are 100 different time series generated by the probabilistic decoder model, as used to compute the mean time series and the uncertainty bands in Fig. 16

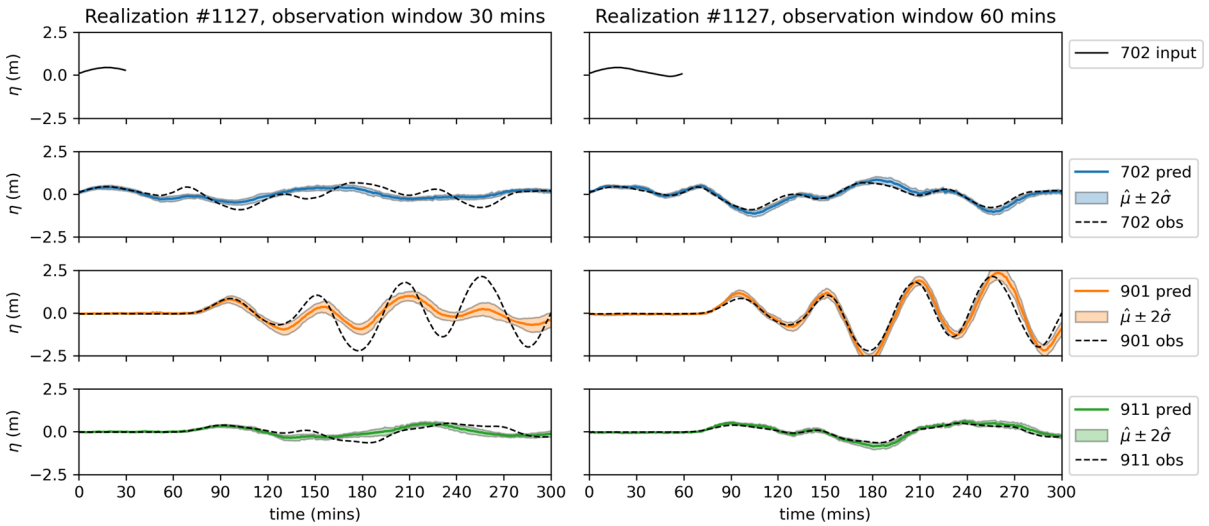

Figure 16
Predicted time series for Realization #1127 using the VAE, based on the observed data shown in the top row for observation window 30 min (left column) or 60 min (right column)

7. *Comparisons and Conclusions*

Figures 6, 9, and 18 show the scatter plots comparing the predicted maximum amplitude of the tsunami at the forecast gauges against the observed values, for the three different approaches we compared. In each case, results are shown for the two gauges 901 and 911, and the results obtained with two different observation windows 30 min and 60 min are presented. The same splitting into training data and test data was used for all methods to provide a fair comparison.

All three approaches are able to make predictions that are generally well correlated with the observations (with the points corresponding to test data lying fairly close to the 45° line). In each case the predictions made based on a 60-min observation window are somewhat better than when only 30 min of observed data is used, as would be expected. This is particularly noticeable for the VAE results of Fig. 18, where $T_{ow} = 30$ min gave a model that tended to

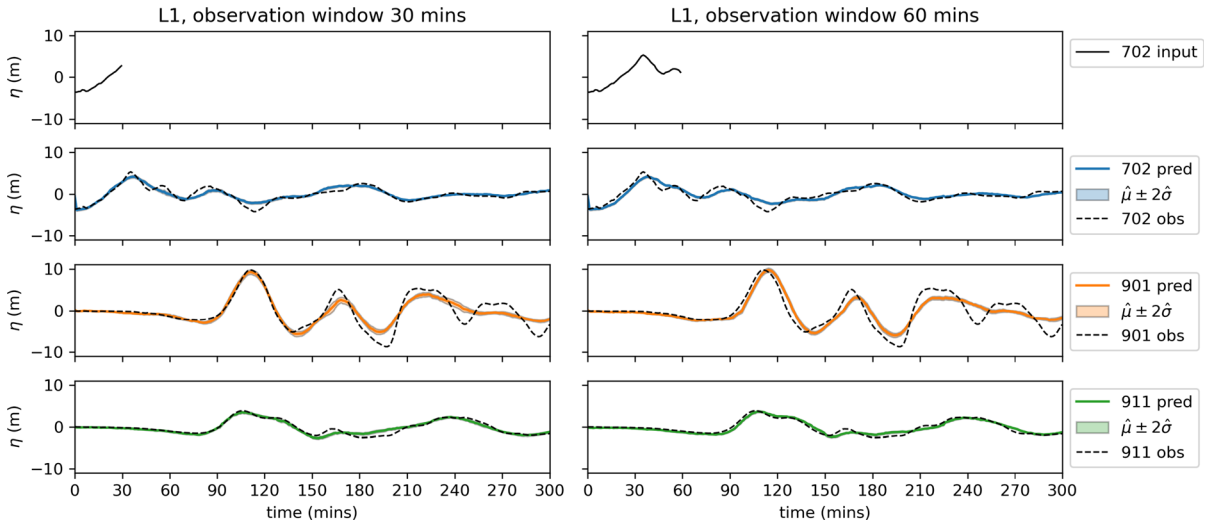

Figure 17
Predicted time series for the L1 scenario using the VAE, based on the observed data shown in the top row for observation window 30 min (left column) or 60 min (right column)

under-predict, resulting in most points lying below the 45° line in the scatter plots.

In order to compare these results more directly, we have computed the average absolute error in peak amplitude over all realizations from the test set, along with the corresponding explained variance score (EVS) as implemented in the `scikit-learn` library (Pedregosa et al., 2011). The EVS is calculated as follows,

$$\mathrm{EVS}(\mathbf{Y}, \widehat{\mathbf{Y}}) = 1 - \frac{\mathrm{Var}\{\mathbf{Y} - \widehat{\mathbf{Y}}\}}{\mathrm{Var}\{\mathbf{Y}\}}. \qquad (30)$$

In our case $\mathbf{Y}$ and $\widehat{\mathbf{Y}}$ are the observed and predicted $\eta_{\max}$, respectively. The closer the computed EVS is to 1.0, the better the model predictions fit the observed values with an EVS of 1.0 being an exact fit. This gives a pair of numbers for each combination of method and T_{ow}, at each forecast gauge. These values are plotted together in Fig. 19. Note that the errors for gauge 901 (half-filled symbols) are much larger than for 911 (solid symbols) because the tsunami amplitude was much larger in Discovery Bay than in Admiralty Inlet, and also perhaps because the response in this shallow location is more nonlinear and harder to predict. So the methods should be compared separately for each gauge.

We can make the following observations, at least for the particular set of methods and hyper-parameters we tested:

- In all cases increasing the observation window from 30 min (blue symbols) to 60 min (red symbols) improves both the average error and the EVS.
- Using the SVM approach on the raw data (squares) generally performed worst, but applying the SVM to features extracted using `tsfresh` (hexagon symbols) gave much better results.
- In all cases tested the DAE (upright triangles) performed best overall. For gauge 901 with 60 min window the VAE (inverted triangles) achieved slightly better results than the VAE in terms of the EVS.

The AE approaches also have the advantage that they produce full time series predictions, and not just estimates of the maximum amplitude. Comparing the time series predictions, e.g. Fig. 11 with Fig. 16 for realization #1127, or Fig. 12 with Fig. 17 for the L1 event, we see that both autoencoder approaches are capable of producing surprisingly good predictions based on short observation windows, particularly for the first wave. This is particularly noteworthy for the L1 event, which is quite different from the realizations generated using the KL expansion that were

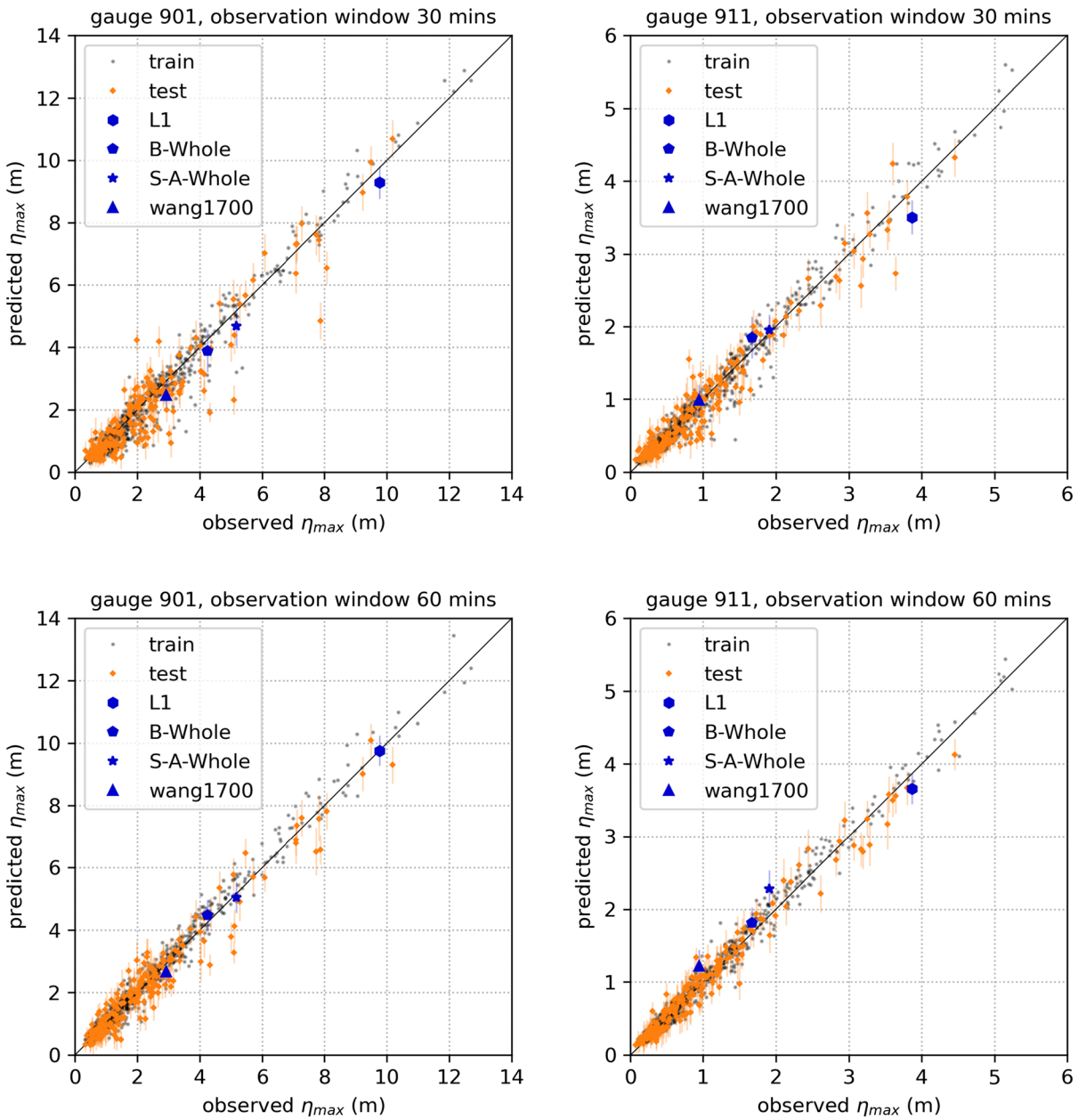

Figure 18
Prediction results using the VAE, as explained in the caption of Fig. 6, with the addition of a vertical bar centered on each test prediction showing 2σ for the uncertainty in the prediction, as explained in Sect. 6

used to train the model, and incorporates a splay fault in addition to slip on the subduction fault surface. For both AE approaches we can produce an ensemble of model results, as described above, and produce a mean forecast along with uncertainty bands. These can be calculated with less work by the VAE than with the DAE approach, which requires separately training each member of the ensemble. However, for

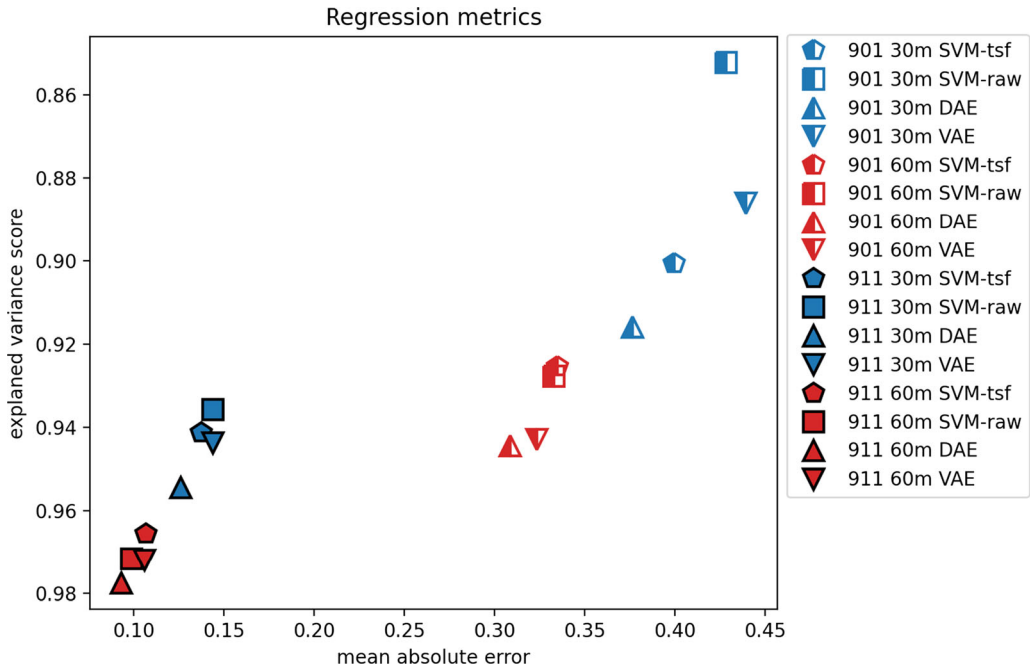

Figure 19
Performance comparison of various models using two metrics, explained variance score (EVS) and mean absolute error over the test set. The EVS is shown increasing downwards since larger values correspond to a better fit. These values are based on absolute errors and hence are larger at Gauge 901 than at 911. The methods compared are SVM-tsf (support vector machine with `tsfresh` features), SVM-raw (support vector machine without `tsfresh` features), DAE (denoising autoencoder) and VAE (variational autoencoder)

all of the models tested it is important to note that, once trained, the model is very quick to apply to new observation data, and so a forecast can be obtained in a fraction of a second. The training process may take much longer, but that would be done in advance of a real event.

We note that the size of the training set used here (959 samples) is relatively modest when compared to the sizes of datasets typically used for training AE models in other applications, often on the order of tens of thousands of samples (LeCun et al., 1998; Li et al., 2017). We expect the results from these models to improve when applied to a larger dataset.

The formulation and training of these forecast models involved careful considerations based on the authors' intuition, especially for the AEs. The architectures were chosen after various iterations of similar models with different number of layers, kernel sizes, activation functions, or latent variable dimensions. Although our experimentation was fairly extensive, it was by no means exhaustive, and so it is possible additional adjustments to the models can result in improvements in performance.

Due to space constraints, we have only shown the full time series predictions for two sample realizations from the test data, #1127 from the fakequakes data set and the L1 event. The website Liu et al. (2021a) includes plots of the time series and predictions for all of the other test problems.

8. *Limitations and Future Research*

The results presented here indicate that these machine learning approaches have the potential to provide excellent forecasts of maximum tsunami amplitudes, and even full time series signals, based on relatively little data at an observation gauge. However, there are potential limitations to this approach and the need for further research on several topics.

We chose a relatively simple forecasting problem for this initial set of tests, since we know that the tsunami entering the Salish Sea passes through the Strait of Juan de Fuca and so a single observation at Gauge 702 may be sufficient. In most tsunami forecasting situations this would not be the case, and collecting multiple observations over a region of the ocean would presumably be necessary in order to characterize the tsunami. The techniques presented in this paper can be easily extended to incorporate short-time observations from multiple gauges as the input data, as might be obtained from a network of seafloor sensors (as already exists on the coast of Japan, for example), or perhaps from multiple DART buoys when forecasting distant tsunamis. We have started to explore these topics and have obtained some promising results that will be presented elsewhere.

Another potential limitation of this work is that we have assumed we have good observations of the surface elevation at the observation gauge. We have not yet explored the robustness of these techniques to noisy observations. Moreover, at locations near the earthquake source, the sea surface elevation is difficult to measure directly from ocean bottom pressure sensors; the acoustic noise generated by the seismic motion can be orders of magnitude larger in amplitude than the changes in hydrostatic pressure that are used to estimate sea surface motion; see e.g. Levin and Nosov (2016) and LeVeque et al. (2018). For nearshore locations like gauge 702 there are potentially other ways to measure surface elevation directly, e.g. high-frequency radar (e.g. Grilli et al. (2016)) or GNSS buoys (e.g. Kato et al. (2018)). Another possibility is to incorporate other data streams into the observations, such as ocean bottom pressure directly, or seismic waveforms from multiple stations, and train the algorithm using this form of data from each hypothetical realization.

In this work we have only tried to forecast time series at gauges. We believe that the same approach can also be applied to predicting two-dimensional maps of maximum onshore inundation over a community of interest, by training the algorithm using such maps from each realization, and we plan to also explore this in the future.

Finally, we note that the VAE could also be used as a generative model to produce additional synthetic tsunami results without requiring additional runs of the modeling software. This could be useful for PTHA or other applications. If we trained the VAE using the full timeseries as both input and output (with the same cost function used here), then the autoencoder produces a set of latent variables representing each run in the training set. The VAE's probabilistic output could be interpreted as a set of new realizations. This is a standard use of autoencoders, but its application in the context of PTHA, for example, would require modification of the model as well as more extensive testing and is beyond the scope of this paper.

Acknowledgements

The authors are grateful to Xinsheng Qin for setting up and performing the GeoClaw simulations used as training and test data. Diego Melgar created the tsunami sources archived at Melgar (2016). Additional earthquake sources used as test data were provided by Kelin Wang, Matthew Sypus, and the NOAA Center for Tsunami Research. Some of the computational experiments were performed using the Greene HPC Cluster at NYU and resources in the Department of Applied Mathematics, University of Washington. The authors also thank the anonymous referees for valuable feedback.

Availability of data and code

All of the data and code used for this work are available through the website Liu et al. (2021a), along with additional plots of time series predictions for other test problems not shown here. The code is also available on Github, and the version used to produce the figures in this paper is permanently archived at Liu et al. (2021b).

Declarations

apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

References

Bernard, E., & Titov, V. (2015). Evolution of tsunami warning systems and products. *Philosophical Transactions of the Royal Society A, 373*(2053), 20140371. https://doi.org/10.1098/rsta.2014.0371.

Chen, P. H., Fan, R. E., & Lin, C. J. (2006). A study on SMO-type decomposition methods for support vector machines. *Transaction on Neural Network, 17*(4), 893–908.

Christ, M., Kempa-Liehr, A., & Feindt, M. (2016). Distributed and parallel time series feature extraction for industrial big data applications. https://arxiv.org/abs/1610.07717

Christ, M., Braun, N., Neuffer, J., & Kempa-Liehr, A. W. (2018). Time series feature extraction on basis of scalable hypothesis tests (tsfresh—a python package). *Neurocomputing, 307,* 72–77.

Clawpack Development Team. (2020). Clawpack software. http://www.clawpack.org, version 5.7.0

Comninou, M., & Dundurs, J. (1975). The angular dislocation in a half space. *Journal of Elasticity, 5*(3–4), 203–216.

Crempien, J. G., Urrutia, A., Benavente, R., & Cienfuegos, R. (2020). Effects of earthquake spatial slip correlation on variability of tsunami potential energy and intensities. *Scientific Reports, 10*(1), 1–10. https://doi.org/10.1038/s41598-020-65412-3.

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. https://doi.org/10.1007/s00024-019-02364-4.

Gao, D., Wang, K., Insua, T. L., Sypus, M., Riedel, M., & Sun, T. (2018). Defining megathrust tsunami source scenarios for northernmost Cascadia. *Natural Hazards, 94*(1), 445–469. https://doi.org/10.1007/s11069-018-3397-6.

Garrison-Laney, C. (2017). Tsunamis and sea levels of the past millennium in Puget Sound, Washington. Thesis, University of Washington, https://digital.lib.washington.edu:443/researchworks/handle/1773/40393

González, F., LeVeque, R.J., Varkovitzky, J., Chamberlain, P., Hirai, B., & George, D.L. (2011). GeoClaw results for the NTHMP tsunami benchmark problems. http://depts.washington.edu/clawpack/links/nthmp-benchmarks

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press, http://www.deeplearningbook.org

Grilli, S.T., Shelby, M., Grilli, A., Gúerin, C.A., Grosdidier, S., & Insua, T. (2016). Algorithms for tsunami detection by high frequency radar: Development and case studies for tsunami impact in British Columbia, Canada. In: The 26th International Ocean and Polar Engineering Conference, International Society of Offshore and Polar Engineers, https://www.onepetro.org/conference-paper/ISOPE-I-16-566

Hastie, T., Tibshirani, R., & Friedman, J. (2001). *The elements of statistical learning. Springer series in statistics*. Springer New York Inc.

Hayes, G. P., Wald, D. J., & Johnson, R. L. (2012). Slab1.0: A three-dimensional model of global subduction zone geometries. *Journal of Geophysical Research Solid Earth*. https://doi.org/10.1029/2011JB008524.

Kato, T., Terada, Y., Tadokoro, K., Kinugasa, N., Futamura, A., Toyoshima, M., et al. (2018). Development of GNSS for a synthetic geohazard monitoring system. *Journal of Disaster Research, 13*(3), 460–471. https://doi.org/10.20965/jdr.2018.p0460.

Kingma, D.P., & Ba, J. (2015). Adam: A method for stochastic optimization. https://arxiv.org/abs/1412.6980

Kingma, D.P., & Welling, M. (2014). Auto-encoding variational bayes. In: *Proceedings of the 2nd International Conference on Learning Representations (ICLR)*, https://arxiv.org/abs/1312.6114

Lakshminarayanan, B., Pritzel, A., & Blundell, C. (2017). Simple and scalable predictive uncertainty estimation using deep ensembles. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems, Red Hook, NY, USA*, NIPS'17, p 6405–6416

LeCun, Y. (1989). Generalization and network design strategies. In R. Pfeifer, Z. Schreter, F. Fogelman, & L. Steels (Eds.), *Connectionism in perspective*. Elsevier.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE, 86*(11), 2278–2324.

LeVeque, R.J., Bodin, P., Cram, G., Crowell, B.W., González, F.I., Harrington, M., Manalang, D., Melgar, D., Schmidt, D.A., Vidale, J.E., Vogl, C.J., & Wilcock, WSD. (2018). Developing a warning system for inbound tsunamis from the cascadia subduction zone. In: *OCEANS 2018 Conference, MTS/IEEE Charleston*, https://doi.org/10.1109/OCEANS.2018.8604709

LeVeque, R., González, F., & Adams, L. (2020). Tsunami hazard assessment of Snohomish county, Washington, Project Report – Version 3. http://staff.washington.edu/rjl/pubs/WA_EMD_Snoho2

LeVeque, R.J., Waagan, K., González, F.I., Rim, D., & Lin, G. (2016). Generating random earthquake events for probabilistic tsunami hazard assessment. Pure Appl Geophys

Levin, B. W., & Nosov, M. A. (2016). *Physics of Tsunamis* (2nd ed.). Springer.

Li, Y., Liu, S., Yang, J., & Yang, M.H. (2017). Generative face completion. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* pp 5892–5900

Liu, C.M., Rim, D., Baraldi, R., & LeVeque, R.J. (2021a). Data and code to accompany this paper. http://faculty.washington.edu/rjl/pubs/MLSJdF2021

Liu, C. M., Rim, D., Baraldi, R., & LeVeque, R. J. (2021b). *Permanent archive of code to accompany this paper.*https://doi.org/10.5281/zenodo.5156748.

Melgar, D. (2016). Cascadia fakequakes waveform data and scenario plots [data set]. journal of geophysical research. https://doi.org/10.5281/zenodo.59943

Melgar, D., Allen, R.M., Riquelme, S., Geng, J., Bravo, F., Baez, J.C., Parra, H., Barrientos, S., Fang, P., Bock, Y., Bevis, M., Caccamise, D.J., Vigny, C., Moreno, M., & Smalley, R. (2016a). Local tsunami warnings: Perspectives from recent large events. Geophys Res Let 43(3):2015GL067100, https://doi.org/10.1002/2015GL067100

Melgar, D., LeVeque, R. J., Dreger, D. S., & Allen, R. M. (2016b). Kinematic rupture scenarios and synthetic displacement data: An example application to the Cascadia Subduction Zone. *J Geophys Res - Solid Earth, 121,* 6658–6674.

Mulia, I. E., & Satake, K. (2020). Developments of Tsunami Observing Systems in Japan. *Frontiers in Earth Science,8,*. https://doi.org/10.3389/feart.2020.00145

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, https://doi.org/10.1029/2020JB019690

Mulia, I. E., Asano, T., & Nagayama, A. (2016). Real-time forecasting of near-field tsunami waveforms at coastal areas using a regularized extreme learning machine. *Coastal Engineering, 109,* 1–8. https://doi.org/10.1016/j.coastaleng.2015.11.010.

NTHMP (2011) Proceedings and results of the 2011 NTHMP Model Benchmarking Workshop. U.S. Department of Commerce/ NOAA/NTHMP; (NOAA Special Report). 436 p., https://nws.weather.gov/nthmp/documents/nthmpWorkshopProcMerged.pdf

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

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., & Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In: Wallach H, Larochelle H, Beygelzimer A, d'Alché-Buc F, Fox E, Garnett R (eds) Advances in Neural Information Processing Systems 32, Curran Associates, Inc., pp 8024–8035

Pathak, D., Krahenbuhl, P., Donahue, J., Darrell, T., & Efros, A.A. (2016). Context encoders: Feature learning by inpainting. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., et al. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research, 12,* 2825–2830.

Rezende, D., Mohamed, S., Wierstra, D. (2014). Stochastic backpropagation and approximate inference deep generative models. In: International Conference on Machine Learning, pp 1278–1286

Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., & Manzagol, P. A. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research, 11,* 3371–3408.

Wang, P. L., Engelhart, S. E., Wang, K., Hawkes, A. D., Horton, B. P., Nelson, A. R., & Witter, R. C. (2013). Heterogeneous rupture in the great Cascadia earthquake of 1700 inferred from coastal subsidence estimates. *Journal of Geophysical Research: Solid Earth, 118*(5), 2460–2473. https://doi.org/10.1002/jgrb.50101.

Williams, H. F., Hutchinson, I., & Nelson, A. R. (2005). Multiple sources for late-Holocene tsunamis at Discovery Bay, Washington State, USA. *The Holocene, 15*(1), 60–73. https://doi.org/10.1191/0956683605hl784rp.

Williamson, A., Melgar, D., & Rim, D. (2019). The Effect of Earthquake Kinematics on Tsunami Propagation. *Journal of Geophysical Research: Solid Earth, 124*(11), 11639–11650. https://doi.org/10.1029/2019JB017522.

Williamson, A. L., Rim, D., Adams, L. M., LeVeque, R. J., Melgar, D., & Gonzaléz, F. I. (2020). A source clustering approach for efficient inundation modeling and regional scale probabilistic tsunami hazard assessment. *Frontiers in Earth Science, 8,* 442. https://doi.org/10.3389/feart.2020.591663.

Witter, R. C., Zhang, Y. J., Wang, K., Priest, G. R., Goldfinger, C., Stimely, L., et al. (2013). Simulated tsunami inundation for a range of Cascadia megathrust earthquake scenarios at Bandon, Oregon, USA. *Geosphere, 9*(6), 1783–1803. https://doi.org/10.1130/GES00899.1.

Zhang, C., Bengio, S., Hardt, M., Recht, B., & Vinyals, O. (2017). Understanding deep learning requires rethinking generalization. https://arxiv.org/abs/2103.10959

(Received April 18, 2021, revised August 3, 2021, accepted August 4, 2021, Published online August 24, 2021)