Deep Learning Seismic Interface Detection using the Frozen Gaussian Approximation

James C. Hateley, Jay Roberts, Kyle Mylonakis, Xu Yang

Department of Mathematics, University of California at Santa Barbara, Santa Barbara, CA 93106

E-mail: `hateleyjc@gmail.com, jayroberts@math.ucsb.edu, kmylonakis@math.ucsb.edu, xuyang@math.ucsb.edu`

October 2018

Abstract. We propose a deep learning algorithm for seismic interface detection, with the neural networks trained by synthetic high-frequency seismograms, efficiently generated by the frozen Gaussian approximation (FGA). The usage of high-frequency data is advantageous due to the fact that it can provide high resolution of substructures. However, generation of sufficient synthetic high-frequency data sets for training neural networks are, in general, computationally challenging to well-known methods. This bottleneck is overcome by a highly scalable computational platform built upon the FGA, which comes from the semiclassical theory and approximates the wavefields by a sum of fixed-width (frozen) Gaussian wave packets.

We first generate the time series of synthetic seismogram data by FGA, which contains accurate traveltime information (from the ray path) but not exact amplitude information (with asymptotic errors not shrinking to zero even at extremely fine numerical resolution). With the synthetic seismogram generated by the FGA, we build network models using an open source API, GeoSeg, developed using Keras and Tensorflow. In particular, networks using encoder-decoder and UNet architectures, despite only being trained on FGA data, can detect an interface with a high success rate from the seismograms generated by the spectral element method (a different numerical method with exact amplitude information at fine numerical resolution). All the tests are done for P-waves (acoustic) and P- and S-waves (elastic), respectively.

Keywords: seismic interface detection, convolutional neural network, high-frequency data, elastic wave equation, frozen Gaussian approximation

Submitted to: *Inverse Problems*

1. Introduction

Various geophysical aspects, e.g., tectonics and geodynamics [1, 18, 19, 29], can be better understood by images of substructures (e.g. locations of seismic interfaces) of the Earth generated by seismic tomography. Neural networks excel at recognizing shapes, patterns and sorting relevant from irrelevant data; this makes them good for image recognition and sorting images. In particular, convolutional neural networks allowed for rapid advances in image classification and object detection [13], and in fact networks have been created for specific tasks, such as, fault detection [2], earthquake detection, *ConvNetQuake* [17], *DeepDetect* [26] and seismic phase arrival times, *PhaseNet* [30].

One obstacle in building a neural network to detect seismic interface is having an ample data set for training. There is constant waveform data being collected by seismic stations across the globe, and generating data by resampling of this seismic data to train a network can be done but is limited by the Nyquist frequency. Seismic data can not be resampled with a Nyquist frequency lower than the highest usable frequency in the data, thus high frequency data is usually preferred as it tends to lead to improved resolution of the substructures. Other obstacles lie within the differences in geological locations, natural phenomenon (e.g. earthquakes) and unnatural phenomenon (e.g. fracking). Using these data sets to train a general neural network is a daunting task, and thus it is natural to use synthetic data for the training of neural networks.

The dominant frequency of a typical earthquake is around 5 Hz [16], which is considered high-frequency thus leading to demanding, and at times, unaffordable computational cost. This makes generation of sufficient synthetic high-frequency data sets for training neural networks computationally challenging to well-known methods. We overcome this difficulty by building a highly scalable computational platform upon the frozen Gaussian approximation (FGA) method for elastic wave equation [8], which comes from the semiclassical theory and approximates the wavefields by a sum of fixed-width (frozen) Gaussian wave packets. The dynamics of each Gaussian wave packet follow ray paths with the prefactor amplitude equation derived from an asymptotic expansion on the phase plane. The whole set of governing equations are decoupled for each Gaussian wave packet, and thus, in theory, each corresponding ODE system can be solved on its own process, which makes the algorithm embarrassingly parallel.

Using synthetic data, Araya-Polo et al. perform inverse tomography via fully connected neural networks with great success in [3] . Their networks use low dimensional features extracted from seismic data as input. Using deeper convolutional neural networks trained on seismogram data may allow the network to pick up on previously unknown signals. The increase in input dimensionality necessitates more sophisticated deep learning techniques than those presented in [3].

In this paper, we propose a deep learning algorithm for seismic interface detection, with the neural networks trained by synthetic high-frequency seismograms. We first generate the time series of synthetic seismogram data by FGA, with which, we build network models using an open source API, GeoSeg, developed using Keras and Tensorflow. We observe that, networks using encoder-decoder and UNet architectures, despite only being trained on FGA data, can detect an interface with a high success rate from the seismograms generated by the spectral element method, which represent true seismic signals when fine time step and mesh size are used in the computation. This is due to the fact that, although FGA does not carry exact amplitude information (with asymptotic errors proportional to the ratio of wavelength over domain size), it contains accurate traveltime information, which the trained neural networks rely more on to detect the location of seismic interfaces. We present the perfomance by considering a simple two-dimensional layered velocity model, with synthetic data generated for both P- and S-waves. We remark that, although it is straightforward in geophysics to identify traveltime as a key factor to detect the interface location for this example, it is not always natural for the trained networks to automatically use this physical intuition.

The paper is outlined as follows: In Section 2, we introduce briefly the mathematical background of FGA and how the synthetic data is generated based on it. In Section 3, we describe the details of the network design. In Section 4, we give two examples to show the performance of networks, and test them by the data generated using the spectral element software package SPECFEM3D[‡]. We make conclusive remarks in Section 5.

2. Frozen Gaussian approximation

We summarize the mathematical theory of FGA in this section; for full exposition and details for the elastic wave equation, see [8]; and for the acoustic wave equation, see [4]. The core idea of the FGA is to approximate seismic wavefields by fixed-width Gaussian wave packets whose dynamics follow ray paths with the prefactor amplitude equation derived from an asymptotic expansion on the phase plane. The ODE system governing the dynamics for each wave packet are decoupled. In theory, each ODE system can be solved on its own process, hence it is embarrassingly parallel. The implementation, as in previous works [8], is with Fortran using message passage interface (MPI). The implementation has a speed up factor of approximately 1.94; hence, doubling the number of cores nearly halves the computational time.

The equation for the forward modeling to generate the training data set we use is the elastic wave equation. Assuming the linear, isotropic Earth model [6], it is,

$$\rho \partial_t^2 \mathbf{u} = (\lambda + \mu)\nabla(\nabla \cdot \mathbf{u}) + \mu \Delta \mathbf{u} + \mathbf{F}, \tag{1}$$

where $\rho, \lambda, \mu, : \mathbb{R}^3 \to \mathbb{R}$ is the material density, the first and second Lamé parameters respectively and $\mathbf{u} : \mathbb{R} \times \mathbb{R}^3 \to \mathbb{R}^3$ is displacement. The differential operators are taken in terms of the spacial variables. Eq. (1) has a natural separation into divergence and curl free components and can also be written as

$$\partial_t^2 \mathbf{u} = c_{\mathrm{p}}^2 \nabla(\nabla \cdot \mathbf{u}) - c_{\mathrm{s}}^2 \nabla \times \nabla \times \mathbf{u} + \mathbf{F}_\rho. \tag{2}$$

‡ https://geodynamics.org/cig/software/specfem3d/

This decomposition represents P-wave, and S-wave respectively with velocities

$$c_{\mathrm{p}}^2(\mathbf{x}) = \frac{\lambda(\mathbf{x}) + 2\mu(\mathbf{x})}{\rho(\mathbf{x})}, \qquad c_{\mathrm{s}}^2(\mathbf{x}) = \frac{\mu(\mathbf{x})}{\rho(\mathbf{x})}, \tag{3}$$

with $c_{\mathrm{p}}(\mathbf{x})$ representing the P-wave speed and $c_{\mathrm{s}}(\mathbf{x})$ representing the S-wave speed.

2.1. The FGA Formulation

We introduce the FGA formula for the elastic wave equation, eq. (2), with initial conditions

$$\begin{cases} \mathbf{u}(0,\mathbf{x}) & = \mathbf{f}^k(\mathbf{x}), \\ \partial_t \mathbf{u}(0,\mathbf{x}) & = \mathbf{g}^k(\mathbf{x}), \end{cases} \tag{4}$$

where the superscript k represents the wavenumber. For a sake of simplicity and clarity, we shall also use the following notations:

- $\mathrm{i} = \sqrt{-1}$: the imaginary unit;
- subscripts/superscripts "p" and "s" indicate P- and S-waves, respectively;
- $\pm$ indicates the two-way wave propagation directions correspondingly;
- $\hat{\boldsymbol{N}}_{\mathrm{p,s}}(t)$: unit vectors indicating the polarized directions of P- and S-waves;
- $\hat{\boldsymbol{n}}_{\mathrm{p,s}}$: the initial directions of P- and S-waves.

The FGA approximates the wavefield $\mathbf{u}^k(t,\mathbf{x})$ in eq. (1) by a summation of dynamic frozen Gaussian wave packets,

$$\begin{aligned} u_{\mathrm{F}}^k(t,\boldsymbol{x}) \approx & \sum_{(\boldsymbol{q},\boldsymbol{p})\in G_{\pm}^{\mathrm{p}}} \frac{a_{\mathrm{p}} \hat{\boldsymbol{N}}_{\mathrm{p}} \psi_{\mathrm{p}}^k}{(2\pi/k)^{9/2}} e^{\mathrm{i}k\boldsymbol{P}_{\mathrm{p}}\cdot(\boldsymbol{x}-\boldsymbol{Q}_{\mathrm{p}}) - \frac{k}{2}|\boldsymbol{x}-\boldsymbol{Q}_{\mathrm{p}}|^2} \delta\boldsymbol{q}\delta\boldsymbol{p} \\ & + \sum_{(\boldsymbol{q},\boldsymbol{p})\in G_{\pm}^{\mathrm{s}}} \frac{a_{\mathrm{s}} \hat{\boldsymbol{N}}_{\mathrm{s}} \psi_{\mathrm{s}}^k}{(2\pi/k)^{9/2}} e^{\mathrm{i}k\boldsymbol{P}_{\mathrm{s}}\cdot(\boldsymbol{x}-\boldsymbol{Q}_{\mathrm{s}}) - \frac{k}{2}|\boldsymbol{x}-\boldsymbol{Q}_{\mathrm{s}}|^2} \delta\boldsymbol{q}\delta\boldsymbol{p}, \end{aligned} \tag{5}$$

with the weight functions

$$\psi_{\mathrm{p,s}}^k(\boldsymbol{q},\boldsymbol{p}) = \int \alpha_{\mathrm{p,s}}^k(\boldsymbol{y},\boldsymbol{q},\boldsymbol{p}) e^{-\mathrm{i}k\boldsymbol{p}\cdot(\boldsymbol{y}-\boldsymbol{q}) - \frac{k}{2}|\boldsymbol{y}-\boldsymbol{q}|^2}\, \mathrm{d}\boldsymbol{y}, \tag{6}$$

$$\alpha_{\mathrm{p,s}}^k(\boldsymbol{y},\boldsymbol{q},\boldsymbol{p}) = \frac{1}{2kc_{\mathrm{p,s}}|\mathbf{p}|^3} \Big(k\boldsymbol{f}^k(\boldsymbol{y}) c_{\mathrm{p,s}}|\boldsymbol{p}| \pm \mathrm{i}\boldsymbol{g}^k(\boldsymbol{y}) \Big) \cdot \hat{\boldsymbol{n}}_{\mathrm{p,s}}. \tag{7}$$

In eq. (5), $G_{\pm}^{\mathrm{p,s}}$ refers to the initial sets of Gaussian center $\boldsymbol{q}$ and propagation vector $\boldsymbol{p}$ for P- and S-waves, respectively. In eq. (7), the "$\pm$" on the right-hand-side of the equation indicate that the $\alpha_{\mathrm{p,s}}^k$ correspond to $(\boldsymbol{q},\boldsymbol{p}) \in G_{\pm}^{\mathrm{p,s}}$. We refer [8] for the derivation, accuracy and explanation of FGA, and only summarize the formulation as follows.

The ray path is given by the Hamiltonian system with Hamiltonian $H(\boldsymbol{Q},\boldsymbol{P}) = \pm c_{\mathrm{p,s}}(\boldsymbol{Q})|\boldsymbol{P}|$. The "$\pm$" give the two-way wave propagation directions; e.g. for the "+"

wave propagation, $(\boldsymbol{q},\boldsymbol{p}) \in G_+^{\mathrm{p,s}}$, the Gaussian center $\boldsymbol{Q}_{\mathrm{p,s}}(t,\boldsymbol{q},\boldsymbol{p})$ and propagation vector $\boldsymbol{P}_{\mathrm{p,s}}(t,\boldsymbol{q},\boldsymbol{p})$ follow the ray dynamics

$$\begin{cases} \dfrac{\mathrm{d}\boldsymbol{Q}_{\mathrm{p,s}}}{\mathrm{d}t} = c_{\mathrm{p,s}}(\boldsymbol{Q}_{\mathrm{p,s}})\dfrac{\boldsymbol{P}_{\mathrm{p,s}}}{|\boldsymbol{P}_{\mathrm{p,s}}|}, \\ \dfrac{\mathrm{d}\boldsymbol{P}_{\mathrm{p,s}}}{\mathrm{d}t} = -\partial_{\boldsymbol{Q}} c_{\mathrm{p,s}}(\boldsymbol{Q}_{\mathrm{p,s}})|\boldsymbol{P}_{\mathrm{p,s}}|, \end{cases} \tag{8}$$

with initial conditions

$$\boldsymbol{Q}_{\mathrm{p,s}}(0,\boldsymbol{q},\boldsymbol{p}) = \boldsymbol{q} \quad \text{and} \quad \boldsymbol{P}_{\mathrm{p,s}}(0,\boldsymbol{q},\boldsymbol{p}) = \boldsymbol{p}. \tag{9}$$

The prefactor amplitudes $\boldsymbol{a}_{\mathrm{p,s}}(t,\boldsymbol{q},\boldsymbol{p})$ satisfy the following equations, where S-waves have been decomposed into SH- and SV-waves,

$$\frac{\mathrm{d}a_{\mathrm{p}}}{\mathrm{d}t} = a_{\mathrm{p}}\Big(\pm\frac{\partial_{\boldsymbol{Q}_{\mathrm{p}}} c_{\mathrm{p}} \cdot \boldsymbol{P}_{\mathrm{p}}}{|\boldsymbol{P}_{\mathrm{p}}|} + \frac{1}{2}\operatorname{tr}\Big(Z_{\mathrm{p}}^{-1}\frac{\mathrm{d}Z_{\mathrm{p}}}{\mathrm{d}t}\Big)\Big), \tag{10}$$

$$\frac{\mathrm{d}a_{\mathrm{sv}}}{\mathrm{d}t} = a_{\mathrm{sv}}\Big(\pm\frac{\partial_{\boldsymbol{Q}_{\mathrm{s}}} c_{\mathrm{s}} \cdot \boldsymbol{P}_{\mathrm{s}}}{|\boldsymbol{P}_{\mathrm{s}}|} + \frac{1}{2}\operatorname{tr}\Big(Z_{\mathrm{s}}^{-1}\frac{\mathrm{d}Z_{\mathrm{s}}}{\mathrm{d}t}\Big)\Big) - a_{\mathrm{sh}}\frac{\mathrm{d}\hat{\boldsymbol{N}}_{\mathrm{sh}}}{\mathrm{d}t}\cdot\hat{\boldsymbol{N}}_{\mathrm{sv}}, \tag{11}$$

$$\frac{\mathrm{d}a_{\mathrm{sh}}}{\mathrm{d}t} = a_{\mathrm{sh}}\Big(\pm\frac{\partial_{\boldsymbol{Q}_{\mathrm{s}}} c_{\mathrm{s}} \cdot \boldsymbol{P}_{\mathrm{s}}}{|\boldsymbol{P}_{\mathrm{s}}|} + \frac{1}{2}\operatorname{tr}\Big(Z_{\mathrm{s}}^{-1}\frac{\mathrm{d}Z_{\mathrm{s}}}{\mathrm{d}t}\Big)\Big) + a_{\mathrm{sv}}\frac{\mathrm{d}\hat{\boldsymbol{N}}_{\mathrm{sh}}}{\mathrm{d}t}\cdot\hat{\boldsymbol{N}}_{\mathrm{sv}}, \tag{12}$$

with the initial conditions $a_{\mathrm{p,sv,sh}} = 2^{3/2}$, and $\hat{\boldsymbol{N}}_{\mathrm{sv}}$ and $\hat{\boldsymbol{N}}_{\mathrm{sh}}$ are the two unit directions perpendicular to $\boldsymbol{P}_{\mathrm{s}}$, referring to the polarized directions of SV- and SH-waves, respectively. With the short-hand notations,

$$\partial_{\boldsymbol{z}} = \partial_{\boldsymbol{q}} - \mathrm{i}\partial_{\boldsymbol{p}}, \qquad Z_{\mathrm{p,s}} = \partial_{\boldsymbol{z}}(\boldsymbol{Q}_{\mathrm{p,s}} + \mathrm{i}\boldsymbol{P}_{\mathrm{p,s}}). \tag{13}$$

For a flat interface $z = z_0$, the wave speeds of the two layers near the interface are assumed to be,

$$c_{\mathrm{p}}(\boldsymbol{x}) = \begin{cases} c_{\mathrm{p}}^{\vee}(\boldsymbol{x}) & z > z_0 \\ c_{\mathrm{p}}^{\wedge}(\boldsymbol{x}) & z < z_0 \end{cases}, \quad c_{\mathrm{s}}(\boldsymbol{x}) = \begin{cases} c_{\mathrm{s}}^{\vee}(\boldsymbol{x}) & z > z_0 \\ c_{\mathrm{s}}^{\wedge}(\boldsymbol{x}) & z < z_0 \end{cases}. \tag{14}$$

As a Gaussian wave packet hits an interface, several of its quantities need to be defined. First, $a_{\mathrm{p,s}}$ and $\boldsymbol{P}_{\mathrm{p,s}}$, are determined by Snell's Law and the Zoeppritz equations [27]. If one denotes $\theta_i, \theta_r, \theta_t$ to be the P-wave incident, reflection and transmission angles, and ϕ_r, ϕ_t to be the SV-wave reflection and transmission angles, respectively, then the Zoeppritz equations read as

$$M\begin{pmatrix} a_{\mathrm{p}}^{\mathrm{re}} \\ a_{\mathrm{s}}^{\mathrm{re}} \\ a_{\mathrm{p}}^{\mathrm{tr}} \\ a_{\mathrm{s}}^{\mathrm{tr}} \end{pmatrix} = \begin{pmatrix} \cos(\theta_r) \\ \sin(\theta_r) \\ \cos(2\phi_r) \\ \cos(2\theta_r) \end{pmatrix} a_{\mathrm{p}}^{\mathrm{in}}, \tag{15}$$

with the matrix M as

$$M = \begin{pmatrix} \cos(\theta_r) & \frac{c_{\mathrm{p}}^{\wedge}}{c_{\mathrm{s}}^{\wedge}} \sin(\phi_r) & \frac{c_{\mathrm{p}}^{\wedge}}{c_{\mathrm{p}}^{\vee}} \cos(\theta_t) & -\frac{c_{\mathrm{p}}^{\wedge}}{c_{\mathrm{s}}^{\vee}} \sin(\phi_t) \\ -\sin(\theta_r) & \frac{c_{\mathrm{p}}^{\wedge}}{c_{\mathrm{s}}^{\wedge}} \cos(\phi_r) & \frac{c_{\mathrm{p}}^{\wedge}}{c_{\mathrm{p}}^{\vee}} \sin(\theta_t) & \frac{c_{\mathrm{p}}^{\wedge}}{c_{\mathrm{s}}^{\vee}} \cos(\phi_t) \\ -\cos(2\phi_r) & -\sin(2\phi_r) & \frac{\rho_2}{\rho_1} \cos(2\phi_t) & -\frac{\rho_2}{\rho_1} \sin(2\phi_t) \\ \sin(2\theta_r) & -(\frac{c_{\mathrm{p}}^{\wedge}}{c_{\mathrm{s}}^{\wedge}})^2 \cos(2\phi_r) & \frac{\rho_2 (c_{\mathrm{p}}^{\wedge} c_{\mathrm{s}}^{\vee})^2}{\rho_1 (c_{\mathrm{p}}^{\vee} c_{\mathrm{s}}^{\wedge})^2} \sin(2\theta_t) & \frac{\rho_2 (c_{\mathrm{p}}^{\wedge})^2}{\rho_1 (c_{\mathrm{s}}^{\wedge})^2} \cos(2\phi_t) \end{pmatrix}, \tag{16}$$

where $\rho_{1,2}$ are the densities for the layers 1 and 2, respectively. Let $\mathbf{N}$ denote the normal to the interface at the point of incidence then $\mathbf{Q}^{\mathrm{in,re,tr}}$ is the Gaussian center at the point of incidence, and $\mathbf{P}^{\mathrm{in,re,tr}}$ corresponds to the propagation vector of incident, reflected and transmitted Gaussian wave packet for either P- or S-waves. $\mathbf{Q}^{\mathrm{in}} = \mathbf{Q}^{\mathrm{re}} = \mathbf{Q}^{\mathrm{tr}}$ and $\mathbf{P}^{\mathrm{re,tr}}$ is updated as follows

$$\mathbf{P}^{\mathrm{tr,re}}_{\mathrm{p,s}} = \mathbf{P}^{\mathrm{in}} + \mathrm{sgn}(\mathbf{P}^{\mathrm{tr,re}}_{\mathrm{p,s}}) \Big(\sqrt{|\mathbf{P}^{\mathrm{in}}| n^{\mathrm{tr,re}}_{\mathrm{p,s}} - \big| |\mathbf{P}^{\mathrm{in}}| - (\mathbf{P}^{\mathrm{in}} \cdot \mathbf{N})^2 \big|} - (\mathbf{P} \cdot \mathbf{N}) \Big) \mathbf{N}, \tag{17}$$

where $n^{\mathrm{tr,re}}_{\mathrm{p,s}}$ denotes the index of refraction for the new respective direction, e.g. $n^{\mathrm{tr}}_{\mathrm{p}} = c^{\vee}_{\mathrm{p}} / c^{\wedge}_{\mathrm{p}}$. Also $Z_{\mathrm{p,s}}$ needs to be updated, requiring use of conservation of level set functions defined in the Eulerian frozen Gaussian approximation formula [14, 25].

$$\begin{aligned} \partial_{\mathbf{z}} \mathbf{Q}^{\mathrm{re,tr}} &= \partial_{\mathbf{z}} \mathbf{Q}^{\mathrm{in}} F, \\ \partial_{\mathbf{z}} \mathbf{P}^{\mathrm{re,tr}} &= \partial_{\mathbf{z}} \mathbf{P}^{\mathrm{in}} W - \frac{|\mathbf{P}^{\mathrm{re,tr}}|}{c(\mathbf{Q}^{\mathrm{re,tr}}) \mathbf{P}^{\mathrm{re,tr}} \cdot \mathbf{N}} \left(\partial_{\mathbf{z}} \mathbf{Q}^{\mathrm{re,tr}} \cdot \nabla c(\mathbf{Q}^{\mathrm{re,tr}}) - \partial_{\mathbf{z}} \mathbf{Q}^{\mathrm{in}} \cdot \nabla c(\mathbf{Q}^{\mathrm{in}}) \right) \mathbf{N}, \end{aligned} \tag{18}$$

F and W are two 3×3 matrices, $F^T = W^{-1}$, and

$$F = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ (\kappa - 1) \frac{p_x}{p_z^{\mathrm{in}}} & (\kappa - 1) \frac{p_y}{p_z^{\mathrm{in}}} & \kappa \frac{p_z^{\mathrm{re,tr}}}{p_z^{\mathrm{in}}} \end{bmatrix}, \quad \text{with} \quad \kappa = \left(\frac{c(\mathbf{Q}^{\mathrm{re,tr}})}{c(\mathbf{Q}^{\mathrm{in}})} \right)^2.$$

2.2. Generation of the Synthetic Data

Large and diverse datasets can reduce generalization error of a network. The data points used for our experiments are synthetic seismograms. Given an initial condition, as in eq. (4), the initial wave packet decomposition can be saved for a variety of tests. This means the same data can be loaded as the parameters vary from data point to data point. If the initial condition is independent of the wave velocities, the same initial wave packet decomposition can be used to generate seismograms with varying velocities, and varying interface depth. Hence for the forward simulation, loading the initial wave packet decomposition, running an ODE solver, and recording the seismograms are the only tasks required. As the ODE system for the FGA is uncoupled for each wave packet, the speed of a single simulation greatly benefits from a parallel implementation. An example of the efficiency and data sets generated are included in Section 4.

3. Network Design

The goal of Full Waveform Inversion (FWI) is to extract wave speed data from seismic data. In its purest form this is a regression type problem and was addressed with fully connected networks in [3]. Our work approaches the problem from a segmentation perspective. We address a simplified version of FWI and attempt to detect subsurface structures by classifying them as regions of low or high wavespeed, thus transforming the regression problem into a segmentation problem. These sorts of segmentation problems have been addressed with great success by CNNs [21].

Semantic segmentation of images is the process of labeling each pixel in an image with a class label for which it belongs. In semantic segmentation problems the correct pixel label map is referred to as the ground truth. In our work the "image" is the one-dimensional slice in the depth direction which we normalize and partition into N bins which act as our "pixels". Each bin is then labeled depending on whether it came from a region of high or low velocity. These velocity regions are our classes. Our work diverges substantially from traditional semantic segmentation of images, as our input is time series data which is then transformed by the network. This is opposed to the traditional case where the input itself is labeled.

The goal of our network is to infer the presence of high and low wavespeed regions and the interfaces between them from seismogram data. The input to the network is $X \in \mathbb{R}^{M\times 3\times r}$. Where M is the number of timesteps and r is the the number of receivers. The output of the network is

$$\mathcal{N}(X) = (p_{ij}), \qquad \begin{array}{c} i \in \{1, ..., M\} \\ j \in \{1, 2\} \end{array}, \tag{19}$$

where p_{ij} is the probability that the i^{th} bin belongs to the j^{th} class. For example, possible output and groundtruth could be

$$\begin{bmatrix} 0.1 & 0.9 \\ 0.2 & 0.8 \\ 0.55 & 0.45 \end{bmatrix}, \quad \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}.$$

Here, at depth indexed by 1, the network believes with 10% probability that this bin is a low speed region and with 90% probability that it is a high speed region, and similarly for the other rows. The accuracy of a given inference is found by taking the row-wise argmax of the matrix, resulting in a column vector, and then evaluating the number of correctly predicted values against the ground truth for the given sample. If one instead takes a row-wise max, one receives a vector whose index represents depth, and whose value represents the probability that the given depth belongs to the speed region predicted by the network. We call this value the heatmap. For instance, the above example has 66% accuracy, and heat map $[0.9, 0.8, 0.55]^{\mathrm{T}}$. The training loss for the neural network is categorical cross-entropy.

3.1. Model Architectures

The models were built using an open source API, GeoSeg[§], we developed using Keras and Tensorflow. GeoSeg allows us to select a UNet, fully convolutional segmentation network, or feed forward NN as a base meta-architecture, using any of residual, dense, or convolutional blocks, with or without batch normalization [20, 21, 10, 11, 12]. GeoSeg also allows for easy hyper parameter selection for model and block architectures, and for training optimizers and parameters. The optimizer for all the models was NADAM using default parameters [5].

The network structures are described by their meta-architecture and their blocks. The meta-architecture describes the global topology of the network and how the blocks interact with each-other. Each block either begins or ends with a tranisition layer that will down or up sample the temporal axis respectively. The block itself preserves the dimensions of its input.

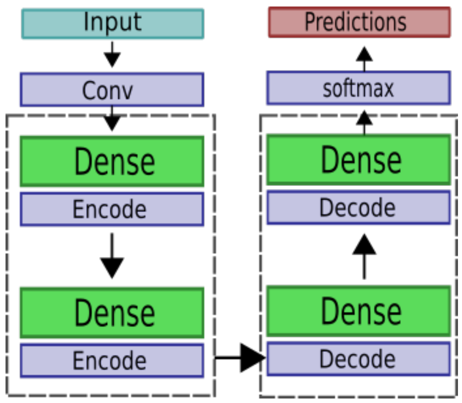
Meta-Architectures. Other than the feed-forward CNN's, all models are variants of Encoder-Decoder Networks similar to SegNet [21]. The input is fed into the encoder branch which down samples the temporal axis by a factor of 2^N where N is the number of layers in the branch. The decoder branch then up samples the temporal dimension by a factor of 2^N. The last layer is a convolutional layer followed by a solftmax which outputs predictions as described above.

GeoDSegB-N refers to a deep network with an N-block long of block type B encoder branch that feeds directly into an N-block long of the same block type decoder branch. GeoDUB-N refers to a UNet architecture from [20]. These architectures have proven highly efficient at image segmentation for road detection [28] and in biomedical applications [20]. These networks feed their input into an encoder branch, bridge block, and then a decoder branch as in GeoSeg. The defining feature of these networks are the "rungs" connecting the encoder and decoder branches (see Figure 1). In this way the network can incorporate both low and high resolution data [20, 28].

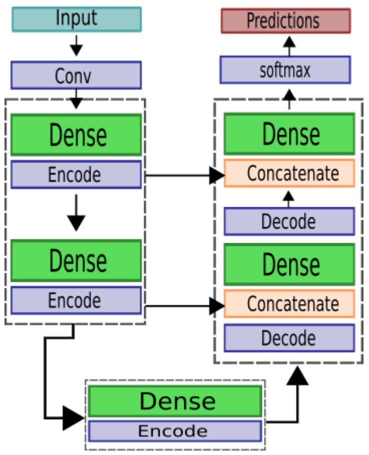
Convolutional Layers. As in [11], we use Batch-Normalization [12] to help smooth training. The layer is broken first into a bottleneck convolution followed by the main convolution. The bottleneck is a convolution which uses a 1x1 kernel to expand the number of feature channels before performing the full convolution. It is suggested in [9, 22] that such a bottleneck can reduce the number of necessary feature maps and so improve computational efficiency. We us Rectified Linear Units (ReLUs) [7] for our activation. Unlike [11], our segmentation problem is one-dimensional, and so we use $k \times 1$ filter kernels as opposed to the traditional $k \times k$. For our networks we chose $k = 3$.

Dense Blocks. Though GeoSeg supports multiple block types all the models reported in this paper use Dense blocks. These are stacks of convolutional layers as shown in Figure 2. The defining features of these blocks, introduced in [11] is that every layer receives input from all previous layers via concatenation. Such architectures have been

§ https://github.com/KyleMylonakis/GeoSeg



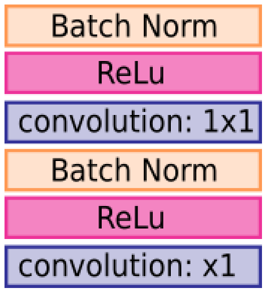
(a) GeoDSegDe-2



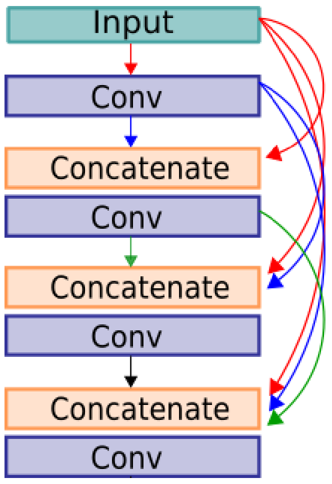
(b) GeoDUDe-2

Figure 1. Meta-architectures of GeoDSegDe-2 (a) and GeoDUDe-2 (b): Notice the presence of rungs and bridge in the UNet architecture.

shown to greatly improve results in image classification while improving computational efficiency [11].



(a) Convolutional Layer



(b) Dense Block

Figure 2. Block compositions of a basic convolutional layer (a) and a corresponding dense block (b)

In [3], Araya-Polo et al. perform inverse tomography via Deep Learning and achieve impressive results. Our model is fundamentally different than GeoDNN in that: GeoDNN is a fully connected network whereas GeoSeg's is fully convolutional, and GeoDNN uses semblance panels from CMP data as features for the network and GeoSeg uses the raw seismograph data. Moreover Araya-Polo et al. address the FWI problem and provide the wave speeds in a two dimensional region and we tackle high and low velocity detection, shifting the problem from regression to segmentation.

4. Experiments and Results

Twenty four models were proposed by considering all choices of meta-architectures, blocks, in two and three block configurations with and without batch normalization. Each model was trained for 3500 epochs with an 80 percent split between training and evaluation data. For every dataset the high velocity region is always below the low velocity region. The training data set only contains time series data generated by the FGA. Separate models were trained for data sets containing only P-waves and those containing both P- and Ss waves. Models without batch normalization had poor evaluation accuracy and loss, and so will not be reported. All models were trained with dropout probability of 50% [23].

Model evaluations were performed with data generated by the FGA and SPECFEM. Notably, the networks are never trained with any SPECFEM data. This was to investigate whether the network was sensitive to the asymptotic error produced by the FGA. We remark in [15, 24], it was shown even small pertubations in input can affect network classification results. Testing on SPECFEM data was also initially proposed to see if neural networks could distinguish between different numerical schemes solutions to problems. This question is not directly addressed in this paper, but remains for future work. All of the models were trained on the Google Cloud Platform with Keras 2.2.2 and Tensorflow 1.10.0 as a backend using a single NVIDIA Tesla V100 GPU.

4.1. results from P-wave data set

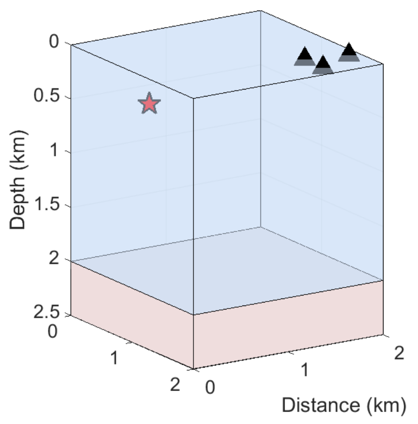
The P-wave data set is generated with a computation domain of $[0,2]\text{km}\times[0,2]\text{km}\times[0,2.5]\text{km}$ with a source centered at $\mathbf{x}_0 = (0.5, 0.5, 0.5)$km with source function

$$f_j^k(\mathbf{x}) = \cos(k(x_j - x_{0,j}))\exp\big(-2k|\mathbf{x}-\mathbf{x}_0|^2\big), \tag{20}$$

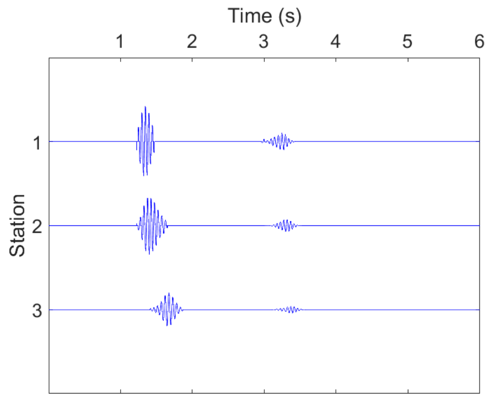
and wavenumber $k = 128$, which is roughly 20.37 Hz. The stations are located on the surface at $S_1 : (1.5, 1.5, 0)$ km, $S_2 : (1.8, 1.5, 0)$ km, $S_3 : (1.6, 1.9, 0)$ km. The interface is a plane, $z = z_0$ that varies from depth 1km to 2.5 km. Above the interface the wavespeed varies from .78 km/s to 1.22 km/s, below the interface the wavespeed varies from 1.29 km/s to 1.56 km/s. As the initial condition is independent of the wavespeed only one wave packet decomposition needs to be computed and saved for all data points to be generated. This saves a tremendous amount of time as only the ODE system needs to be solved for varies wavespeeds and interface heights. The data is generated on the cluster, *knot*, at the center for scientific computing at UC Santa Barbara[||], using 64 processes with a 4th order Runge-Kutta solver for the ODE system. 804672 total beams are used and each data point is generated in approximately 2.5 minutes. This is compared to

|| http://csc.cnsi.ucsb.edu/clusters/knot

SPECFEM3D which takes is approximately 45 minutes to generate a data point. Using the FGA, there are 7790 total data points generated. Each data point is a (6000,3,3) tensor. Prior to training, we further down sample the temporal dimension by a factor of 25 and normalize the amplitude of the seismogram data. All the networks were trained with a mini-batch size of 256 examples.



(a) Computational Domain



(b) Seismogram

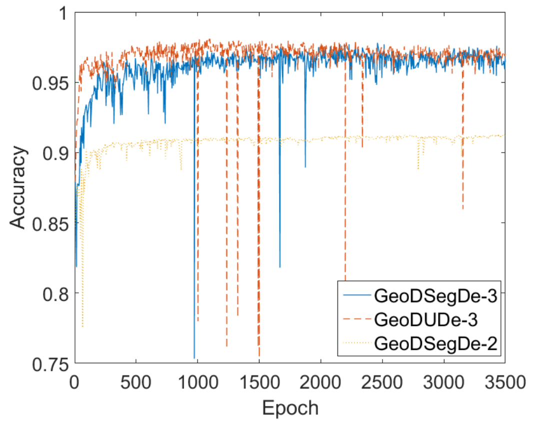
Figure 3. P-wave set up. Wavenumber k=128. (a) The is source located at (.5,.5,.5)km as a star and the 3 receivers located on the surface, the interface presented is at a depth of 2km. (b) A typical data point, a collection of 3 seismograms from the forward simulation using the FGA.

The only models which had above a 90% evaluation accuracy were GeoDSegDe-2, GeoDSegDe-3, and GeoDUDe-3. The training and evaluation loss and accuracy are shown in Fig. 4. GeoDUDe-3 had the best evaluation accuracy with 96.97% on the FGA dataset, with GeoDSegDe-3 being a very close second. The evaluation accuracy between two and three branch layer models is significant. Visualizations of results for GeoDUDe-3 are shown in Figures 6, 5. The reported losses and accuracies for all the networks are shown in Table 1.

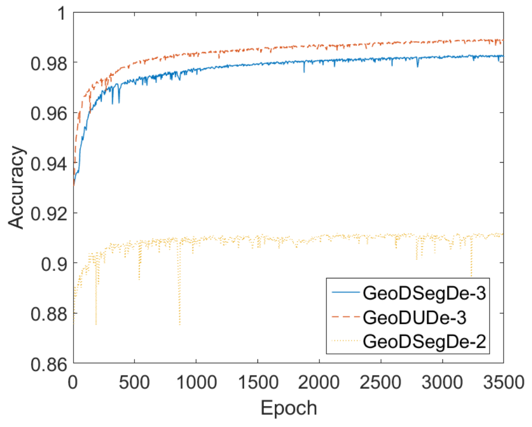
Table 1. P-Data Network Comparisons.

Network	Eval. Acc.	Eval. Loss	Train. Acc.	Train. Loss	SEM Acc.
GeoDSegDe-2	91.22 %	0.1957	91.19 %	0.1935	90.32 %
GeoDSegDe-3	96.42 %	0.08457	98.24 %	0.05284	93.83 %
GeoDUDe-3	96.97 %	0.09700	98.86 %	0.03388	94.29 %

The various networks were also evaluated on 100 samples generated by SPECFEM3D instead of the FGA. On this dataset each neural network had only slightly worse evaluation accuracy than on data generated by the FGA with a maximum difference between the evaluation accuracies of the datasets being 2.68%.

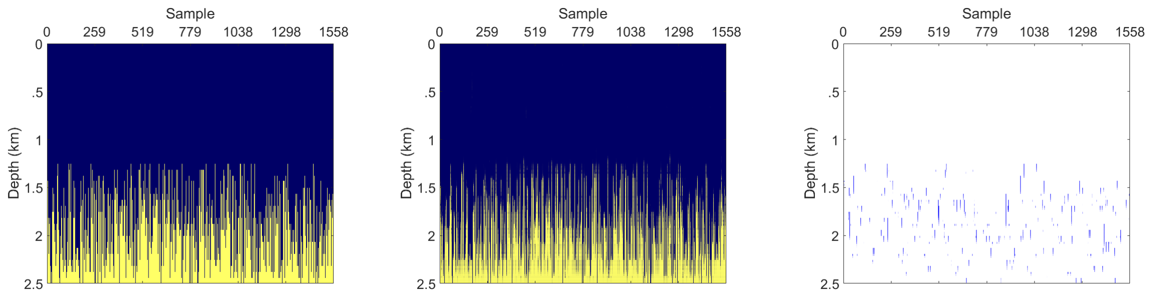


(a) Evaluation Accuracy

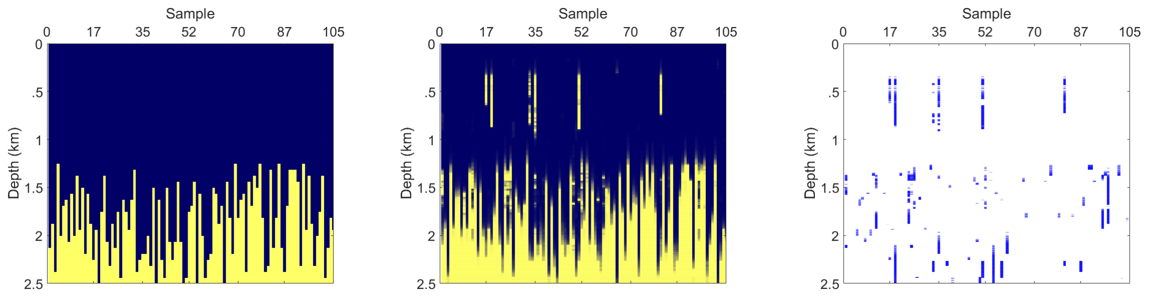


(b) Training Accuracy

Figure 4. P-wave training results. The evaluation dataset for this figure only contains data generated by the FGA. Accuracy is computed as the total number correct classifications.



(a) FGA: Actual (b) FGA: Predicted (c) FGA: Difference



(d) SEM: Actual (e) SEM: Predicted (f) SEM: Difference

Figure 5. GeoDUDe-3: P-wave predictability. Each column of pixels represents a sample. The value of each pixel describes whether the material at the depth corresponding to that pixel's column belongs to either the high or low wavespeed region.

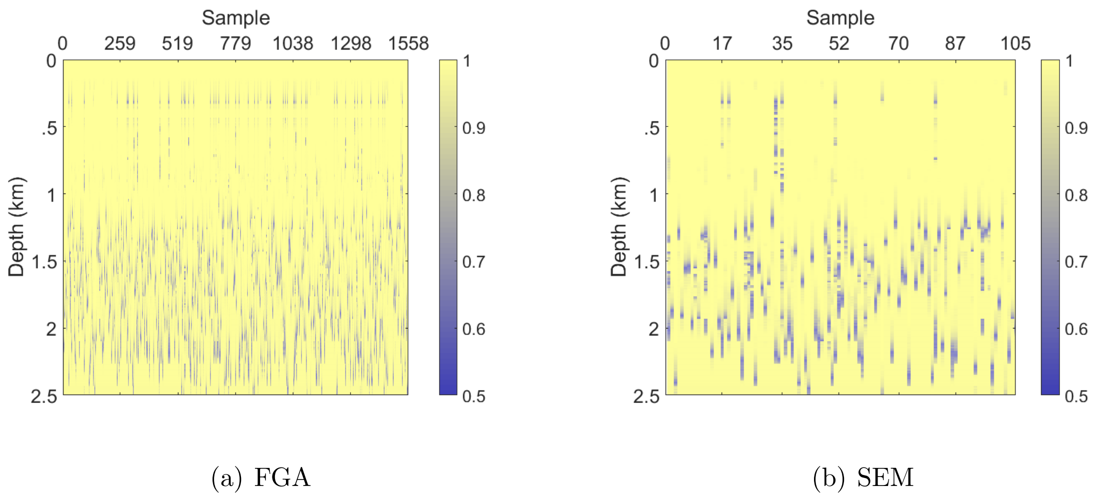


Figure 6. GeoDUDe-3: P-wave heat-map distribution comparison. Regions of low confidence correspond to areas where an interface is likely. Recall the heatmap is a sample is the vector of probabilities that a depth indicated by the index belongs is correctly classified.

4.2. results from P,S-wave data set

The P,S-wave dataset is generated with a computation domain of $[0,2]$ km$\times[0,2]$ km$\times[0,3]$ km with a source centered at $\mathbf{x}_0 = (0.5, 0.5, 0.5)$ km with source function based off the Greens function:

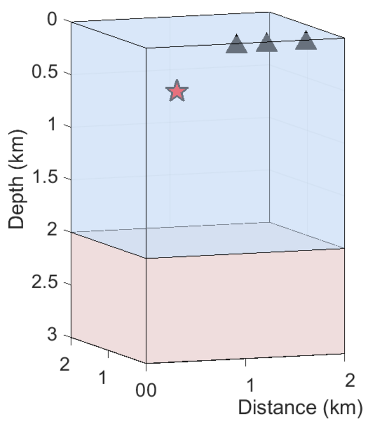
$$\begin{aligned} f_j^k(\mathbf{x}) = \sum_{i=1}^{3} \frac{(x_i - x_{i,0})(x_j - x_{j,0})}{4\pi\rho c_{\mathrm{p}}^2 r^3} F_j(t_0 - r/c_{\mathrm{p}}) + \\ \frac{r^2\delta_{ij} - (x_i - x_{i,0})(x_j - x_{j,0})}{4\pi\rho c_{\mathrm{s}}^2 r^3} F_j(t_0 - r/c_{\mathrm{s}}) + \\ \frac{3(x_i - x_{i,0})(x_j - x_{j,0}) - r^2\delta_{ij}}{4\pi\rho r^5} \int_{r/c_{\mathrm{p}}}^{r/c_{\mathrm{s}}} sF_j(t_0 - s)\ \mathrm{d}s, \end{aligned} \tag{21}$$

$F_j(t) = \cos(kt)\exp(-2kt^2)$, δ_{ij} is the Kronecker delta, $t_0 = 2\sqrt{1/k}$ and wavenumber is set to $k = 32$, this is approximately 5.09 Hz. The stations lie in a plane and are located just below the surface at $S_1 : (1.1, 0.5, 0.1)$ km, $S_2 : (1.4, 0.5, .1)$ km, $S_3 : (1.8, 0.5, 0.1)$ km. The interface is a plane, $z = z_0$ that varies from depth 1km to 2km. Above the interface c_{p} varies from 0.75 km/s to 1.10 km/s, below the interface c_{p} varies from 1.12 km/s to 1.48 km/s and we fix $c_{\mathrm{s}} = c_{\mathrm{p}}/1.7$ (corresponding the case $\lambda = \mu$ roughly). There are a total of 6,400 data points in the P,S-wave dataset. Each data point is a (2048,3,3) tensor. Prior to training each example is down-sampled along the temporal axis by a factor of 8. To help convergence, the number of feature channels of the neural networks for the P,S-Wave dataset were increased by a factor of 4 as compared to the networks trained on the P-Wave dataset. Each network used a mini-batch training size of 256. Similarly to the P-wave dataset, 100 additional samples were generated using SPECFEM3D.

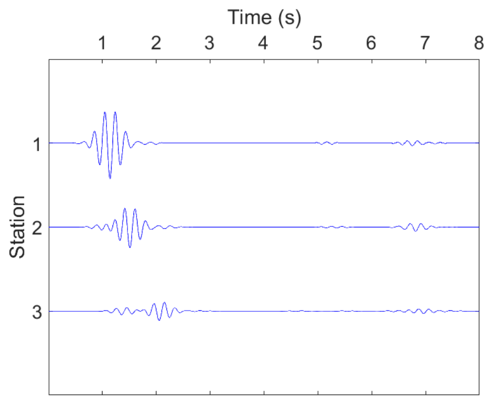
In general the UNet architecture had higher evaluation accuracy compared to the segmentation networks. The most successful model was GeoDUDe-2, with 98.26 % evaluation accuracy on FGA data, and 97.55 % evaluation data on the SPECFEM data.

Due to the significant increase in the number of parameters in the networks using PS-wave data, we find that the evaluation accuracy goes down for deeper networks. In particular, GeoDUDe-3 performed significantly worse with only a 92.34 % evaluation accuracy, especially compared to the same network architecture on the P-wave dataset. This is likely due to overfitting of the data causing an increase in generalization error.

Similarly to the P-wave dataset, evaluation accuracies on SPECFEM3D data are only marginally worse than their FGA counterparts, with a max difference of 1.76% between the datasets.



(a) Computational Domain

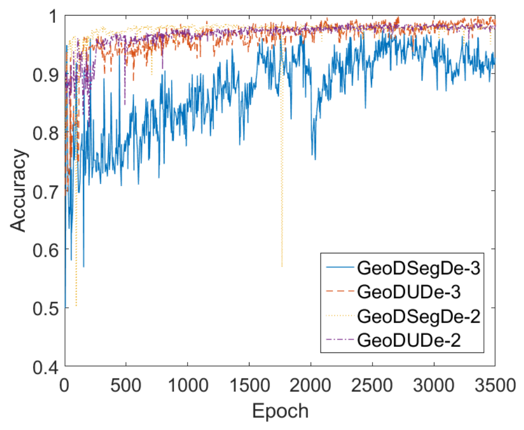


(b) Seismogram

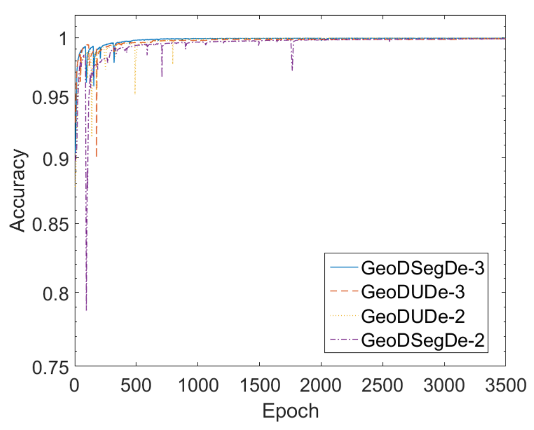
Figure 7. P,S-wave setup. Wavenumber k=32. (a) The is source located at (.5,.5,.5) as a star and the 3 receivers located on the surface, the interface presented is at a depth of 2km. (b) A typical data point, a collection of 3 seismograms from the forward simulation using the FGA.

Table 2. P,S-Data Network Comparisons.

Network	Eval. Acc.	Eval. Loss	Train. Acc.	Train. Loss	SEM Acc.
GeoDSegDe-2	97.74 %	0.1742	99.89 %	0.00375	97.44 %
GeoDSegDe-3	92.34 %	0.4290	99.95 %	0.0019789	90.58 %
GeoDUDe-2	98.26 %	0.1650	99.97 %	0.0015729	97.55 %
GeoDUDe-3	97.64 %	0.1975	99.90 %	0.0026064	96.47 %

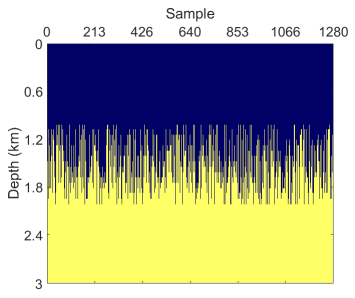


(a) Evaluation Accuracy

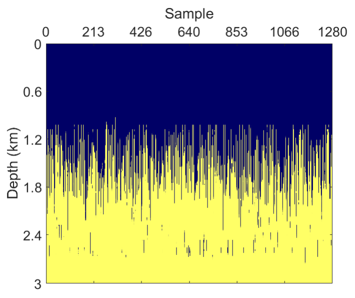


(b) Training Accuracy

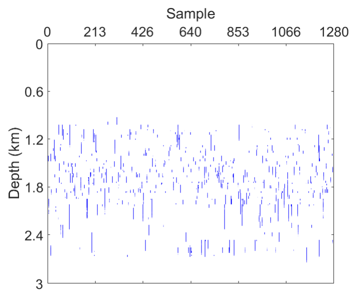
Figure 8. PS-wave training results. Wavenumber k=32: The evaluation dataset for this figure only contains data generated by the FGA. Accuracy is computed as the total number correct classifications.



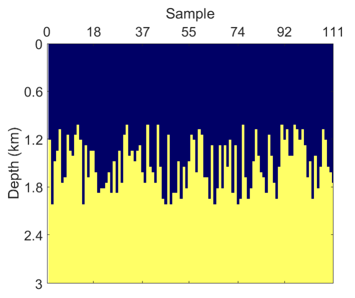
(a) FGA: Actual



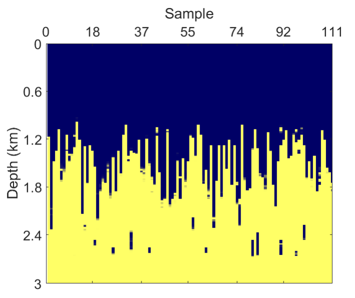
(b) FGA: Predicted



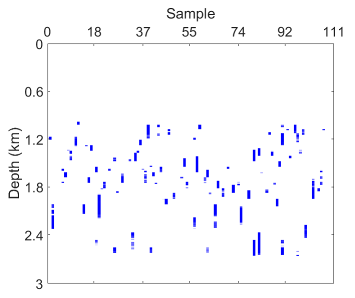
(c) FGA: Difference



(d) SEM: Actual

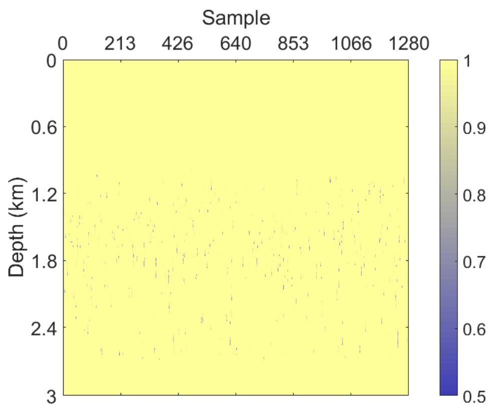


(e) SEM: Predicted

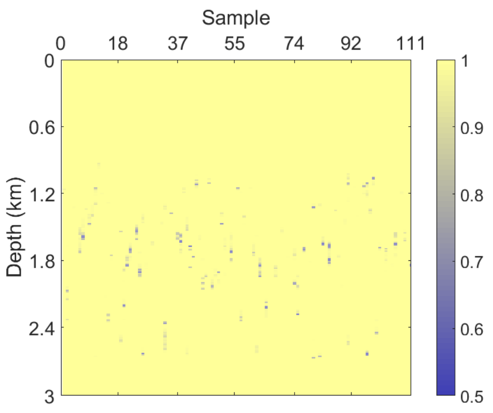


(f) SEM: Difference

Figure 9. GeoDUDe-2: P,S-wave predictability. Each column of pixels represents a sample. The value of each pixel describes whether the material at the depth corresponding to that pixel's column belongs to either the high or low wavespeed region.



(a) GeoDUDe-2:FGA



(b) GeoDUDe-2:SEM

Figure 10. GeoDUDe-2: P,S-wave heat-map distribution comparison. Regions of low confidence correspond to areas where an interface is likely.

5. Conclusions and future work

The use of the FGA to generate large amounts of seismic data provides a quick way to generate labeled synthetic data for statistical learning of the inverse tomography problem. Casting the inverse problem as a segmentation problem resulted in high evaluation accuracy networks for piecewise constant two-layer models on both FGA and SEM datasets. The Encode/Decode fully convolutional and UNet architectures with Dense blocks displayed superior accuracy compared to simpler network architectures. However, deeper networks did not necessarily outperform their shorter counterparts. Yet, all models exhibited good invariance of prediction in regard to which numerical method was used to generate the dataset, as the FGA and SEM exhibit the same traveltime information. Having a network independent of numerical method is important, and the FGA can help to train such a network as it generates synthetic seismic data that carries the correct traveltime information of the real-world data.

The success of the networks on this piecewise constant simple interface act as a stepping stone to tackle more complicated and realistic geological models. By developing the API GeoSeg, available at https://github.com/KyleMylonakis/GeoSeg, it is easy to implement neural networks designed for more general segmentation problems of seismogram data than those discussed in this paper. Immediate future directions to be explored are multi-layer models, piecewise linear models, and non-linear interface models. A long term goal is to develop a neural network trained on synthetic seismic data capable of making inferences from real seismic data.

Acknowledgement

We acknowledge support from the Center for Scientific Computing from the CNSI, MRL: an NSF MRSEC (DMR-1720256) and NSF CNS-0960316. The work was partially supported by the NSF grant DMS-1818592. XY also thank Professors Haizhao Yang and Kui Ren for useful discussions.

References

[1] K. Aki and W. Lee. Determination of the three-dimensional velocity anomalies under a seismic array using first P arrival times from local earthquakes 1. A homogeneous intial model. *J. Geophys. Res.*, 81:4381–4399, 1976.

[2] M. Araya-Polo, T. Dahlke, C. Frogner, C. Zhang, T. Poggio, and D. Hohl. Automated fault detection without seismic processing. *The Leading Edge*, 36(3):208–214, 2017.

[3] M. Araya-Polo, J. Jennings, A. Adler, and T. Dahlke. Deep-learning tomography. *The Leading Edge*, 37(1):58–66, 2018.

[4] L. Chai, P. Tong, and X. Yang. Frozen Gaussian approximation for 3-D seismic wave propagation. *Geophysical Journal International*, 208(1):59–74, 2017.

[5] T. Dozat. Incorporating nesterov momentum into adam. 2016.

[6] A. M. Dziewonski and D. L. Anderson. Preliminary reference earth model. *Physics of the earth and planetary interiors*, 25(4):297–356, 1981.

[7] X. Glorot, A. Bordes, and Y. Bengio. Deep sparse rectifier neural networks. In *AISTATS*, 2011.

[8] J. C. Hateley, L. Chai, P. Tong, and X. Yang. Frozen gaussian approximation for 3-d elastic wave equation and seismic tomography. *Geophys. J. Int.*, Submitted.

[9] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[10] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[11] G. Huang, Z. Liu, and K. Q. Weinberger. Densely connected convolutional networks. *CoRR*, abs/1608.06993, 2016.

[12] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.

[13] Y. LeCun, Y. Bengio, and G. E. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[14] J. Lu and X. Yang. Frozen Gaussian approximation for general linear strictly hyperbolic systems: Formulation and Eulerian methods. *Multiscale Model. Simul.*, 10:451–472, 2012.

[15] S. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard. Universal adversarial perturbations. *CoRR*, abs/1610.08401, 2016.

[16] H. Nakamichi, H. Hamaguchi, S. Tanaka, S. Ueki, T. Nishimura, and A. Hasegawa. Source mechanisms of deep and intermediate-depth low-frequency earthquakes beneath iwate volcano, northeastern japan. *Geophysical Journal International*, 154(3):811–828, 2003.

[17] T. Perol, M. Gharbi, and M. Denolle. Convolutional neural network for earthquake detection and location. *Science Advances*, 4(2), 2018.

[18] N. Rawlinson, S. Pozgay, and S. Fishwick. Seismic tomography: A window into deep Earth. *Phys. Earth Planet. Inter.*, 178(3-4):101–135, 2010.

[19] B. Romanowicz. Seismic tomography of the Earth's mantle. *Annu. Rev. Earth Planet. Sci.*, 19:77–99, 1991.

[20] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015.

[21] E. Shelhamer, J. Long, and T. Darrell. Fully convolutional networks for semantic segmentation.

2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 3431–3440, 2015.

[22] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. A. Riedmiller. Striving for simplicity: The all convolutional net. *CoRR*, abs/1412.6806, 2014.

[23] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[24] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *CoRR*, abs/1312.6199, 2013.

[25] D. Wei and X. Yang. Eulerian gaussian beam method for high frequency wave propagation in heterogeneous media with discontinuities in one direction. *Commun. Math. Sci.*, 10:1287–1299, 2012.

[26] Y. Wu, Y. Lin, Z. Zhou, D. C. Bolton, J. Liu, and P. Johnson. Deepdetect: A cascaded region-based densely connected network for seismic event detection. *IEEE Transactions on Geoscience and Remote Sensing*, pages 1–14, 2018.

[27] O. Yilmaz. *Seismic Data Analysis: Processing, Inversion, and Interpretation of Seismic Data.* Society of Exploration Geophysicists, 2001.

[28] Z. Zhang, Q. Liu, and Y. Wang. Road extraction by deep residual u-net. *CoRR*, abs/1711.10684, 2017.

[29] D. Zhao. Tomography and dynamics of Western-Pacific subduction zones. *Monogr. Environ. Earth Planets*, 1:1–70, 2012.

[30] W. Zhu and G. C. Beroza. PhaseNet: A Deep-Neural-Network-Based Seismic Arrival Time Picking Method. *ArXiv e-prints*, Mar. 2018.