Extracting Dispersion Curves From Ambient Noise Correlations Using Deep Learning

Xiaotian Zhang, Zhe Jia, Zachary E. Ross, and Robert W. Clayton

***Abstract*— We present a machine learning approach to classify the phases of surface wave dispersion curves. Standard frequency-time analysis (FTAN) analysis of seismograms observed on an array of receivers is converted into an image, of which each pixel is classified as fundamental mode, first overtone, or noise. We use a convolutional neural network (U-Net) architecture with a supervised learning objective and incorporate transfer learning. The training is initially performed with synthetic data to learn coarse structure, followed by fine-tuning of the network using approximately 10% of the real data based on human classification. The results show that the machine classification is nearly identical to the human picked phases. Expanding the method to process multiple images at once did not improve the performance. The developed technique will facilitate the automated processing of large dispersion curve data sets.**

***Index Terms*— Convolutional networks, deep learning, dispersion curves, surface waves.**

I. Introduction

THE inversion of surface waves has become a standard method for determining the near-surface shear velocity. One reason for this is that surface waves can be relatively easily extracted from ambient noise correlations and hence are not dependent on a suitable distribution of earthquakes. Another reason is that surface waves only require coverage over a 2-D plane and not a 3-D volume, which is what would be required for body waves (S-waves), which are difficult to extract from ambient noise correlations. This becomes important when dealing with dense seismic arrays that typically have a short deployment time.

The method usually consists of three steps, with the first being the determination of dispersion curves, which are measurements of the velocity as a function of frequency. Once this is done, a tomographic method is used to convert these line measurements into maps of the phase or group velocity as a function of frequency [1]. The final step is to then convert velocity as a function of frequency at each (x, y)-point, to velocity as a function of depth, thus making a 3-D model of the subsurface velocity. In this article, we will focus on applying machine learning to the first step in the process—determining the dispersion curves.

Manuscript received February 5, 2020; revised April 29, 2020; accepted April 29, 2020. Date of publication May 25, 2020; date of current version November 24, 2020. This work was supported in part by NSF/EAR under Grant 1520081. *(Corresponding author: Zachary E. Ross.)*

The authors are with the California Institute of Technology, Pasadena, CA 91125 USA (e-mail: zross@gps.caltech.edu).

Color versions of one or more of the figures in this article are available online at https://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TGRS.2020.2992043

A commonly used procedure to extract the dispersion curves is the frequency-time analysis (FTAN) method [2], [3], in which the correlated signal between two stations is filtered by a sequence of zero-phase narrowband filters to determine the travel time of the surface waves as a function of frequency. Knowing the separation distance of the stations allows these measurements to be converted into phase velocity. If the envelope of the signal is used instead of the seismograms themselves, then the group velocity is determined.

At a given frequency, there may be a number of modes present which correspond to different eigenfunctions (dependences with depth). The fundamental mode is the slowest mode with the overtones increasing in velocity as the eigenfunctions penetrate deeper in depth. A key part of determining the dispersion curve is "picking" the travel time or equivalently the velocity since the distance is known. This is similar to the problem of picking P- and S-waves in determining earthquake locations, but here, the various surface wave modes need to be classified for the inversion process. We typically pick the fundamental and first-overtone modes, and occasionally the second-overtone modes if it can be seen. The more that are picked, the better the resolution of the resulting shear velocity model.

Picking the dispersion curves is very labor-intensive, particularly when dealing with dense arrays. The motivation for automating this procedure is not only the large volume of data that are now available (an example of which is shown in Fig. 1) but also the increased precision that is now required because of the density of stations. The process can be machine-assisted by defining target zones for the curves, but the output needs to be checked and adjusted because of spurious noise within these zones. Developing an automatic method to determine the dispersion curves is the subject of this article.

In recent years, deep learning has become the state of the art in numerous areas of artificial intelligence, which has quickly translated into major advances within seismology. Such applications include detection and picking of seismic waves [4], [5], signal denoising [6], and phase association [7]. Recently, Hu *et al.* [8] developed an end-to-end convolutional network approach for predicting 1-D velocity profiles directly from surface wave dispersion curves. These problems can all be cast as supervised learning objectives and benefit from the wealth of labeled data sets that exist in the seismological community. They bear structural similarities to that of dispersion curve picking, motivating the application of deep neural networks.

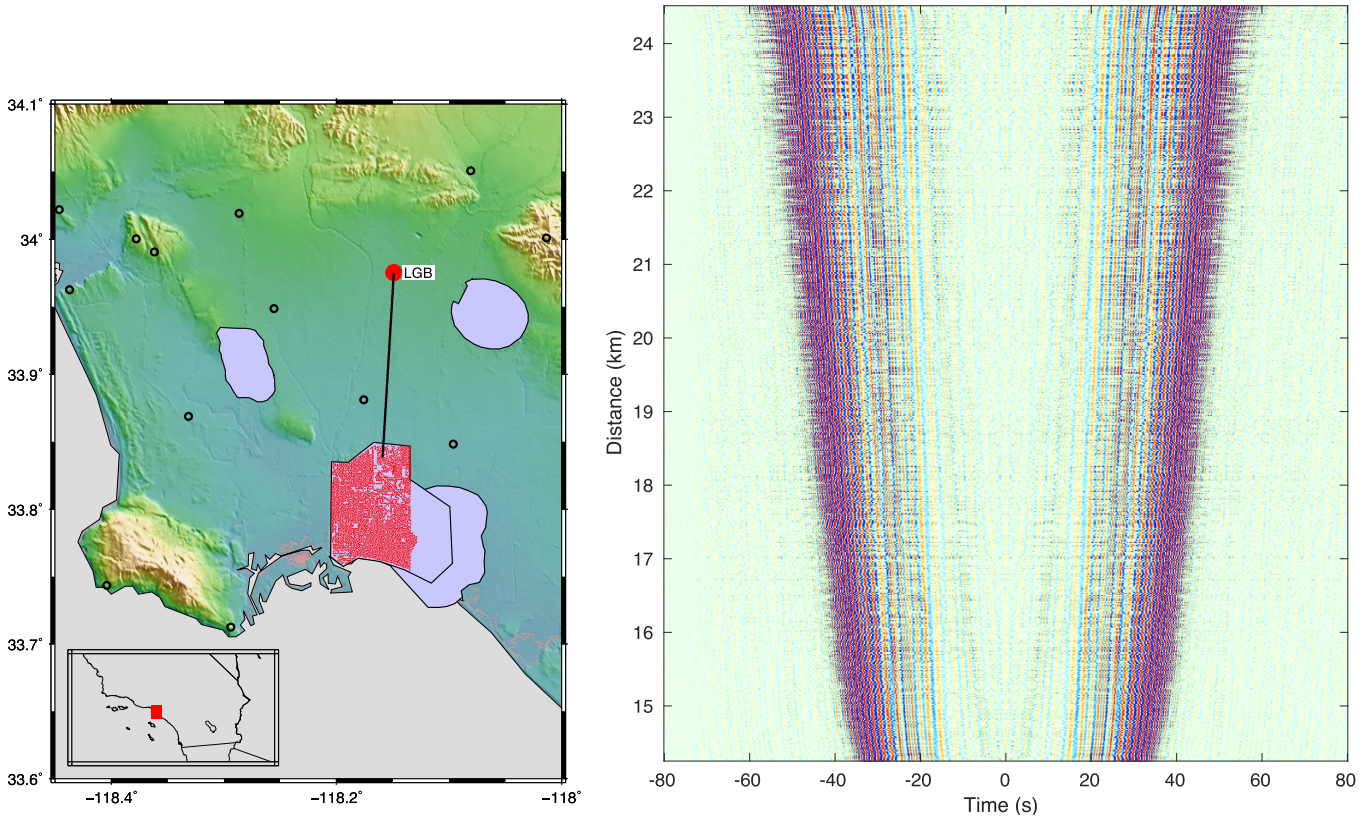


Fig. 1. Location and correlations. (Left) Location of the industry arrays (shaded polygons) and the SCSN broadband stations (black circles). (Right) Long Beach array with 5300 stations (red dots) and the broadband station LGB (big red dot) that are used to create the correlations. The correlations are done for approximately 500 h and are bandpassed 0.1–3.0 Hz.

In this article, we develop a deep learning approach to the dispersion picking problem with the goal of classifying tentative picks as fundamental mode, first overtone, or noise. Our approach uses deep convolutional networks to learn a low-dimensional representation of the data that can be used for pixelwise segmentation of dispersion curves. We show that our approach can reliably and efficiently classify picks, which will greatly facilitate the automated processing of large seismic data sets.

II. Data: Real and Synthetic

In this article, we use data from a temporary dense seismic network of 5340 stations in Long Beach, CA, USA [9] that were originally used for an exploration survey conducted by an oil company. The broadband station LGB is part of the permanent earthquake monitoring array in the region (Southern California Seismic Network) and is cross-correlated with this array to form 5340 station pairs, from which we wish to determine the dispersion curves. The geometry of the array and a sample cross correlation are shown in Fig. 1. The FTAN method was applied to each correlation pair for a range of frequencies between 0.2 and 5 Hz to construct the images of dispersion curves. The bandpass filter is a Gaussian filter $H(\omega) = \exp(-\alpha(\omega - \omega_0)^2/\omega_0^2)$ where we set the filter parameter $\alpha = 25$ for a compromise between the narrowband assumption and filtering robustness. These were then handpicked (labor-intensive) to create a set of labeled dispersion curves.

To limit the amount of labeled data required to train a model to pick dispersion curves, we designed an approach to generate realistic synthetic training data, with the ultimate goal of applying a second stage of real data training to the model. To generate synthetic dispersion curves, we started with a 1-D layered velocity model (Fig. 2) that matched the average dispersion curve for the region of the survey. This function was then perturbed both in velocities and layer thicknesses by a random amount up to 10% of the original velocity function, forming an ensemble of different velocity models. The random variations on the layer thicknesses and velocities for this ensemble were drawn from a uniform distribution that centered at the starting 1-D velocity model. For each instance, the dispersion curves were determined by a numerical solution of the eigenproblem [3], [10]. The resulting curves were then altered by random variations of up to ±2.5% in the frequencies and velocities and by adding random noise to the curves. In total, 100 000 synthetic curves were generated.

For our real data set of 5340 station pairs, we set aside 4340 of the curve images to be used as a testing set, and the rest (1000) were used as a training set. In a production environment, we would hand-label only the 1000 training images and allow the algorithm to determine all remaining 4340 correlation pairs (testing set) without intervention. Here, we need to hand-label even the testing set in order to assess our model's final performance. Note that we train on real data after we have a model trained on synthetic data.

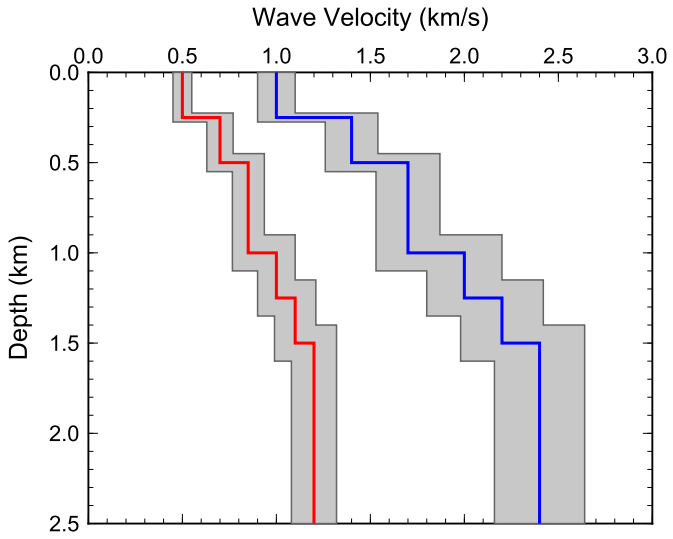


Fig. 2. Velocity model. The P- and S-wave models used to construct the synthetic training set are shown along with the variations in velocity and layer depths.

The synthetic and real dispersion curves are first preprocessed into image representations to make them suitable for the U-Net architecture. Initially, each curve is a collection of points with (frequency, velocity, and amplitude) values such as shown in Fig. 3. Each point also has an associated label from one of the three classes: fundamental mode, first overtone, and noise. We detrend each curve in the frequency–velocity domain and transform the frequency axis into the logarithmic domain to get a more reasonable distribution—see Fig. 3 to see that points are more concentrated on the left of the plot (shorter period). Then, we create a 64×64 pixel grayscale image with pixel values between 0 and 255 representing the amplitude (see Fig. 3). To map individual points to the pixels, we treat each pixel as a discretized bin and fill its grayscale value as the amplitude of points that fall into the bin after integer-rounding. Note that the rounding process occurs after logarithmic transform for the x-axis (periods) and detrending for the y-axis (group velocities). The detrending process utilizes the points from all of the stations, including the test stations, but does not use any ground-truth data. These variable transforms are done to more evenly spread points across the final grayscale image, thus reducing computational requirements. The ground-truth labels for each point of the dispersion curves are mapped to individual pixels by the same process. In the case that two points on the dispersion curve map onto the same pixel, and their labels conflict, we choose to label both as noise since curve fitting is more severely impacted by erroneous points than by missing points.

III. Methods

A. Overview

Our approach to picking dispersion curves uses deep convolutional networks in a supervised manner to perform pixelwise segmentation of the images. It consists of two main steps: 1) a U-Net architecture is trained first on the entirely synthetic data set to learn coarse features and 2) the best model is then fine-tuned to the limited amount of real data using a two-stage training approach. In the following, we describe each of these steps in detail.

The use of convolutional networks is well-motivated by the structure of our data, as the dispersion curve images exhibit a spatially coherent geometric structure [11]. Our problem is set up as one of the fully supervised image segmentation since we have pixelwise labels for all images. The model used in this study is the U-Net architecture [12], which is a deep convolutional network that has been successful for image segmentation tasks. In particular, the network applies a series of convolution and pooling layers to an input image to learn a sparse representation of it and then applies a series of transpose convolution layers to finally output an image with the same lateral dimensions as the input. The depth of the output image is equal to the number of classes, which in our case is 3: fundamental mode, first-overtone, and noise.

Fig. 4 provides a summary of the model used in this study. The network takes in a $64 \times 64 \times 1$ image and outputs a stack of three images of equivalent dimension, with a softmax activation function applied to the outputs. In our case, the output of the neural network is a $64 \times 64 \times 3$ array, with each pixel having three probabilities—noise, fundamental mode curve, and first overtone curve—associated with it. An example of an input image and the corresponding labels are shown in Fig. 5 (top).

To overcome the discretization error due to the 64×64 pixelization of the images, we utilize the picks on pixels by finding the corresponding closest frequency–time energy peaks in the original FTAN maps. To do this, we first use our mapping from "point in period-velocity space" to "pixel coordinate," and then for each pixel that we labeled as "fundamental pick" or "first-order pick," we find the corresponding period-velocity space coordinate and search for the largest amplitude point in the vicinity.

The reason that we do a search is that if we were to use the 64×64 picks directly without searching, we would end up with a maximum of only 64 possible values on both the x- and y-axes, which is an unnecessary source of error. This could be ameliorated by using 128×128 or more pixels, but this would increase the computational requirements by a factor of 4 and would never reach the granularity of finding the original coordinates.

B. Convolutional Neural Network Training

Starting the training process using data from a simulation (a sim2real approach) avoids the need for extensive human labeling. We set aside 10% of the synthetic images for model validation purposes and train the network on the remainder using the Adam optimizer [13] using the minibatches of size 32. After each epoch of training, we check the model's performance on the 10 000 unseen images in the validation set. If performance does not improve for 3 epochs in a row, we save the model after the best epoch.

Next, we proceed to fine-tune the best model on the synthetic data to the real Long Beach data using a two-stage training approach. Of the 5340 images from the Long Beach

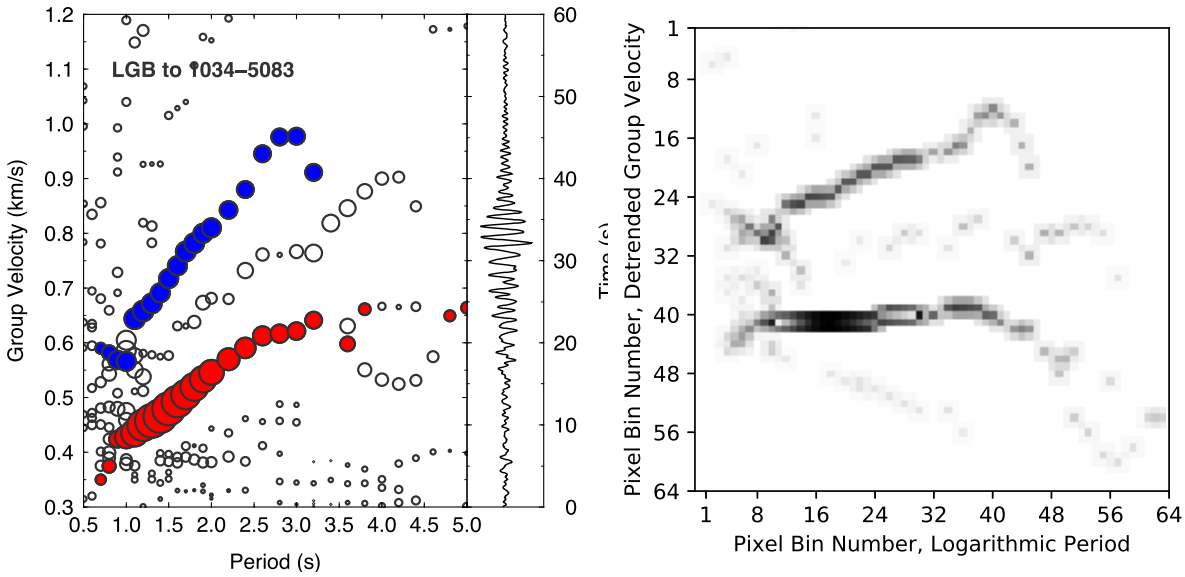


Fig. 3. Example of a dispersion curve generated from FTAN for the path shown in Fig. 1 that has been hand-labeled. Red circles: fundamental picks. Blue circles: first-order picks. Empty circles: noise. The size of each circle is proportional to its amplitude. The waveform is the raw data from a single row of Fig. 1 (right), rotated by 90°. (Right) Pixel representation with logarithmic period axis and the group velocity's linear trend removed.

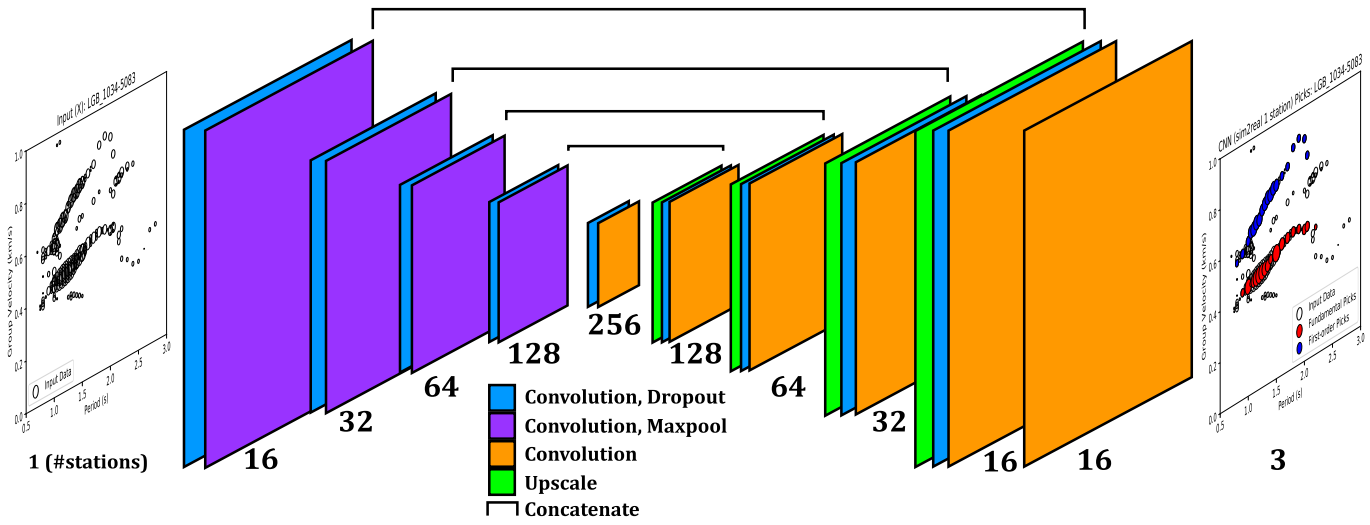


Fig. 4. Cartoon flowchart of how our data is processed throughout this article. In the plots, the [X, Y, Size] axes represent [Period (s), Group Velocity (km/s), Amplitude], respectively. The center of the figure represents the convolutional neural network (CNN) structure that we employed. The goal of this article is to take in a noisy plot and pick out points inside it that belong to the fundamental (red) and first-order (blue) overtones while discarding everything else as noise.

data set, we take a random subset of 1000 images for use in two-stage training. Of these 1000 images, 100 are used for validation (checking when to stop training the model) and 900 are used for updating model weights, which results in a 90%–10% train–validation split. For these 1000 images, we followed the same training procedure as with the synthetic data. The remaining 4340 stations are reserved for evaluation of our models—we pretend that we have no access to their correct labels until after the final model is saved, as they are used only to determine whether our method is suitable for real usage.

Example output predictions are shown in Fig. 5 along with the raw feature input and labels. It is clear that the model performs well for this examined image, correctly recovering nearly all fundamental and first overtone picks. Quantitative performance results on the validation set are provided in Fig. 6, where precision and recall are computed for each of the three classes. The noise class has the highest precision and recall of the three classes (>99%), which probably reflects the fact that the composition of the training data set is heavily skewed toward the noise. The fundamental and first-order modes have around 99% and 98% median precision, respectively, demonstrating that the model can accurately classify individual pixels. The median recalls for these classes are about 95% and 94%, respectively. The application of ML to this problem can substantially reduce the human labor in analyzing surface wave data.

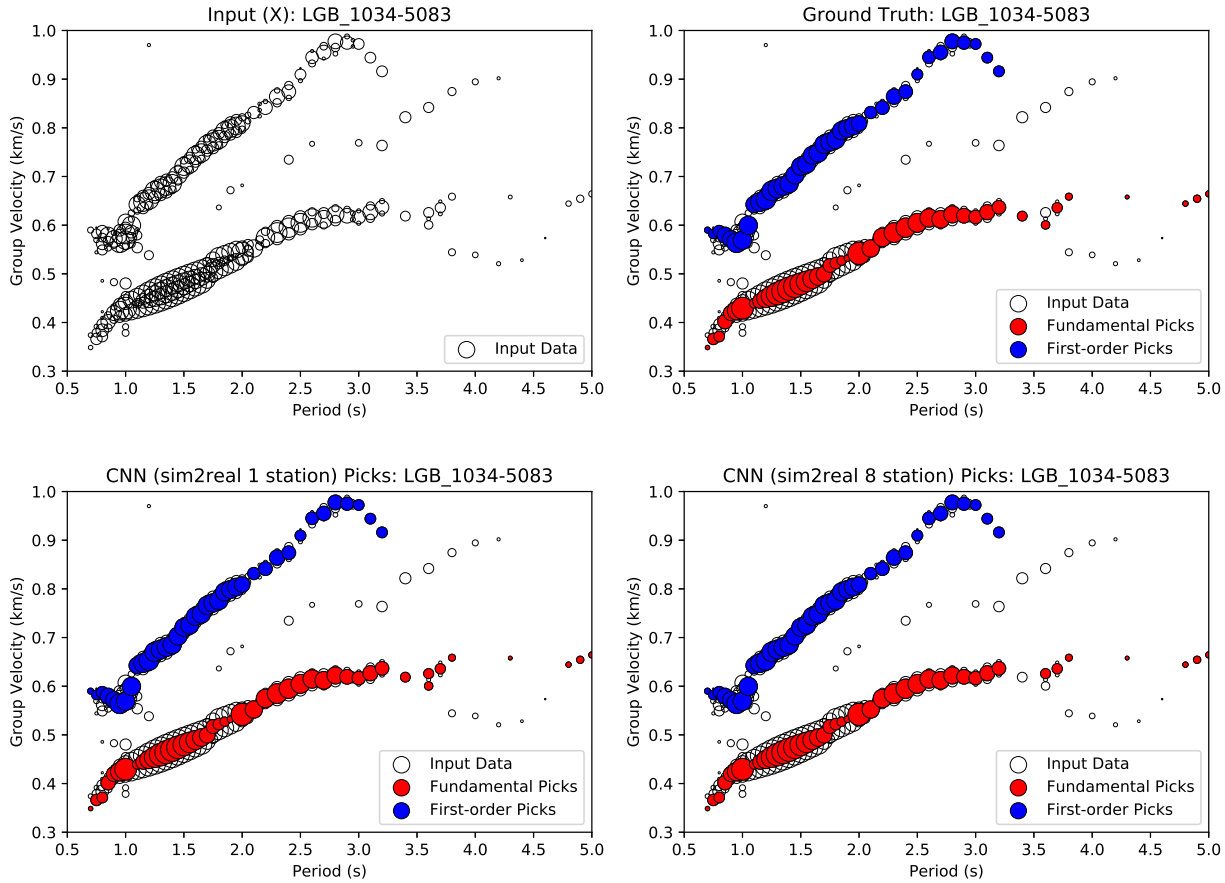


Fig. 5. Results for different algorithms. (Top left) Input data, which are converted to grayscale pixels. (Top right) Handpicked classification of the dispersion curves. (Bottom left) Convolutional network prediction of labels using only one station input. (Bottom right) Results using gight stations input. Note that this station's Ground Truth panel is never seen by the CNN models—it is shown just for comparison and evaluation.

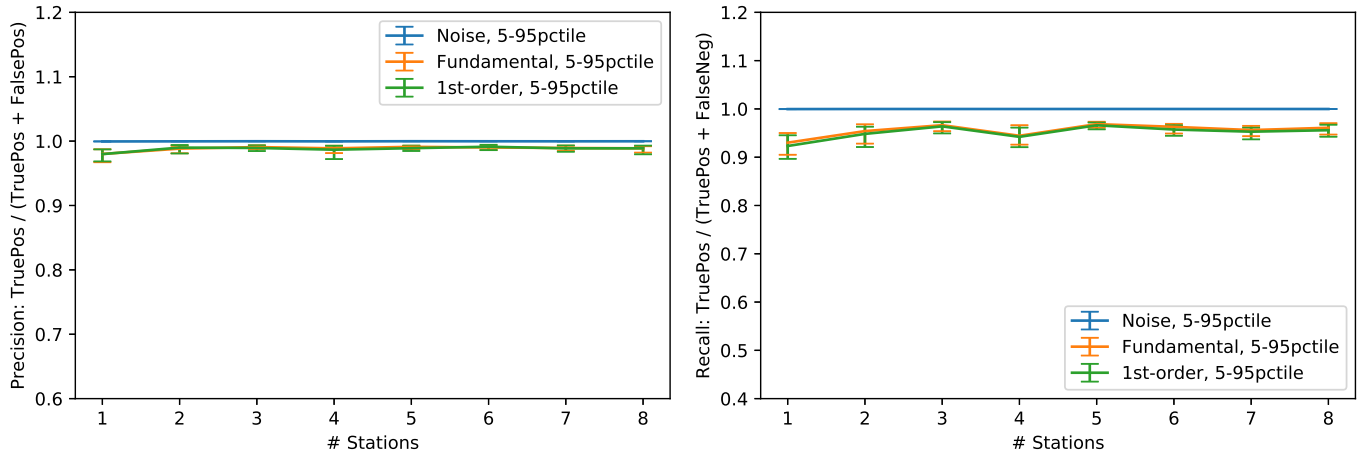


Fig. 6. Precision and recall. Median precision and recall compared with the number of stations presented to the neural network simultaneously. $N = 23$ (top 10% of $N = 239$ models in validation error) of synthetically trained and then Long-Beach trained models. Variations are due to the changes in random seed for the model training, which changes the weight initializations and the order of training data shown.

C. Multistation Input

The analysis described earlier was done for each station pair. We also explored the possibility of including neighboring stations into the feature set to better facilitate separation of genuine signal from noise. While the velocity structure may vary between different pairs, here, we assume that these changes are small enough that the general characteristics from one dispersion curve to another are overall similar. The motivation is to use all of the available information together to make a decision, rather than examining one station pair at a time.

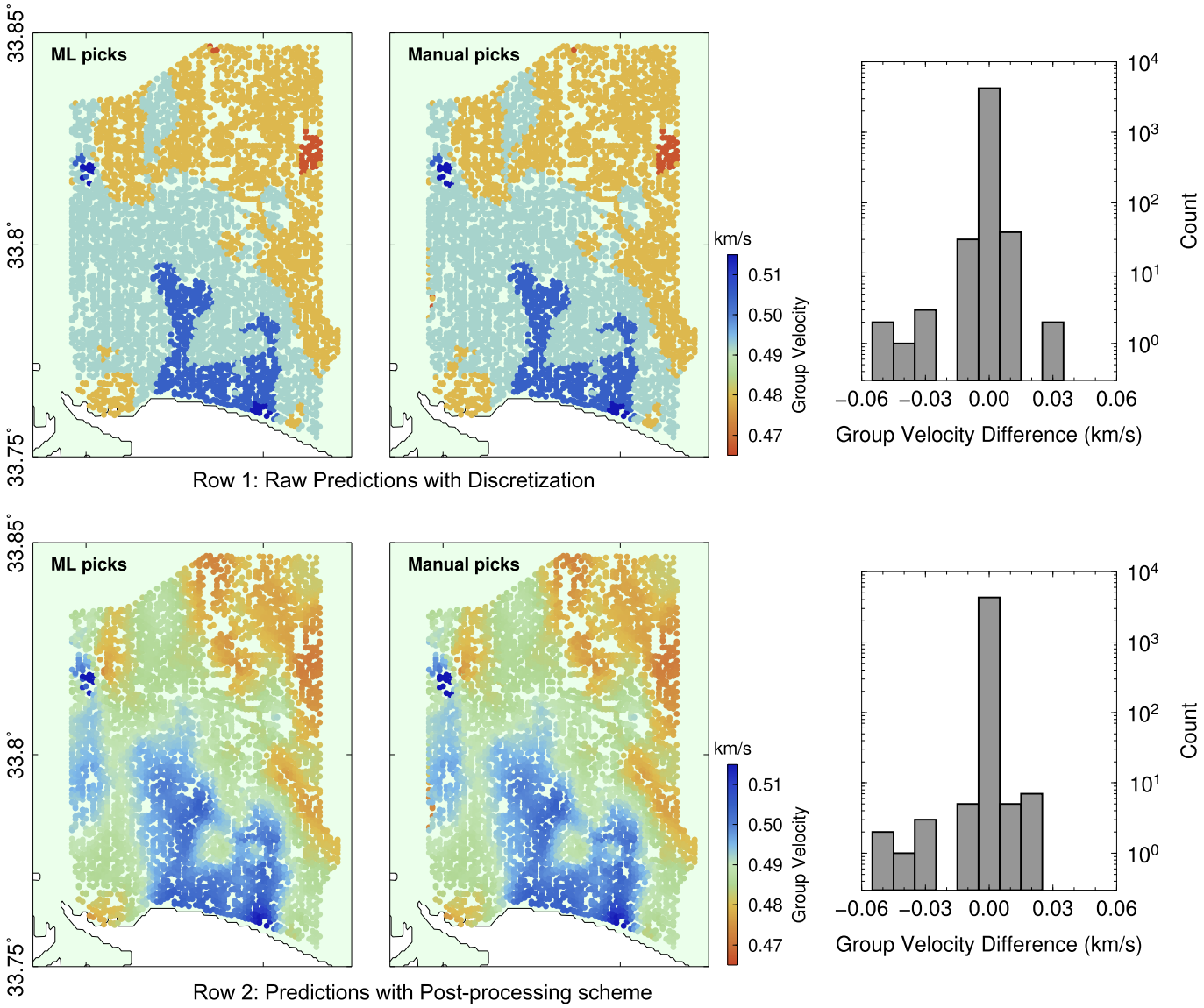


Fig. 7. Results for Long Beach Example. Shown are the group velocity maps for the Long Beach data using the machine learning approach (column 1) and manual approach (column 2). Row 1 contains the discretized predictions, in which group velocity can take on only 64 values—these are what the model trained on. Row 2 uses a postprocessing scheme (Methods, Section-A) to search for the original point, so no discretization error exists. The third column is a histogram of differences between the ML and manual picks. The two types of processing produce nearly identical results indicate that the machine learning approach is working as desired.

To do this, we include the images for K nearest neighbor stations by concatenating them in the depth dimension to create a 3-D input volume. Thus, the inputs are $64 \times 64 \times K$. We repeat the entire training procedure starting from generating synthetic data, including the real data training. An example of a model using $K = 8$ is shown in Fig. 5, which can be compared with the results for the $K = 1$ model seen previously. Fig. 6 shows the median performance while increasing K from 1 to 8. After accounting for the variability introduced by the stochastic nature of the training process (examining the 107 best training runs out of 215 total), we find that the performance does not improve significantly when adding in additional stations. We hoped that noise seen in one station might not be seen in a neighboring station, so a neural network might be able to combine multichannel information to determine that this idiosyncratic noise is indeed noise. However, this unfortunately is not the case.

IV. Discussion

The approach developed in this article provides a tool to train deep neural networks to perform dispersion curve picking using a hybrid simulation + real data scheme. The two-stage training enables the neural network to learn coarse features from the simulation data that are also present in the real data, with the benefit that as much simulated data can be generated as needed. By then fine-tuning the model to the real data, the network learns finer scale features that are unique to these data while only needing a relatively small amount of it. Here, we showed that just 1000 images are enough to adapt the initial model to the real data, although if more is available, the performance may improve further. This type of

sim2real approach will likely be relevant to other problems in seismology where labeled data can be initially obtained from simulations.

We expect that the developed approach will be valid for many types of arrays and velocity structures, although it is likely that the model will need to be trained separately for each region or data set of interest. This is because we expect the velocity structure and network geometry to vary significantly between data sets, making it harder to generalize. Our approach, being designed to use synthetic data for the first stage of training, helps to minimize the amount of new (real) training data needed.

The approach in this article was designed to automate the picking process. This differs from the approach of Hu *et al.* [8], which performs an end-to-end prediction of a velocity depth profile given an input image. One of the distinguishing features between these approaches is that our method focuses only on making precise picks. These can then be used with standard geophysical inverse techniques, providing greater control over the velocity model solutions, via regularization, as well as direct interpretability of the obtained velocity model since the physical equations are used in solving the inverse problem mapping picks to velocity.

We attempted many tweaks to the model hyperparameters as well as the randomization of workflow (same synthetic model + many real models, many synthetic models + many real models, varying learning rates, varying image dimensions, fixed versus variable terminating epoch numbers, zero-padded empty channels versus completely deleted empty channels, changing the probability threshold for picks, and so on) but failed to see any meaningful increase in performance as we increased station count. Therefore, it appears that there is no need to use more than one station for the proposed method, which also requires the least amount of training data.

We also tried to frame the problem as one of sequential classification, treating the picks as a sequence rather than converting to an image and using bidirectional gated recurrent units (similar to Ross *et al.* [4]). The sequences of floating-point tuples (period, group velocity, and amplitude) directly extracted from the FTAN analysis were sorted by amplitude and presented to the neural network. This approach did not perform nearly as well as the convolutional network approach described earlier.

As for the generalization ability of the model, performance on very geographically sparse data has not been tested. The method described here is designed for dense seismic networks in a localized area, such as the arrays shown in Fig. 1, where there are over 16 000 stations. It is not likely to work every well for sparse measurements in an area with a rapidly varying subsurface structure. In the case of multiple geographic clusters of stations, it is advised to hand-label training station data that are randomly sampled from each cluster, as opposed to directly using the model trained entirely from one cluster for a different cluster. This method mainly serves to reduce human work, rather than a pretrained model that will work on every geographic location without further training. If the stations are sparsely scattered, performance is expected to degrade relative to our dense array scenario.

To use our model's picks and see its geographical importance, we plotted in Fig. 7 a group velocity map for our test set (no training stations are plotted) and place it side-by-side with the same plot generated from our manual picks. Each dot represents a station's group velocity at the $T = 1.5$ s period, measured over the entire path from the LGB broadband station to the local array station—see Fig. 1 for the path. We compare the two versions by taking a difference and plot the results on the same figure as well. We can see that the differences between the model and the manual versions are small and few in number, which agrees with the precision/recall data from Fig. 6.

V. Conclusion

We have developed a machine learning method for extracting dispersion curves from velocity–frequency images. The procedure was training with a combination of synthetic examples and labeled real data. Testing on real data shows that the method works with a median per-class precision of at least 98% and a per-class recall rate of at least 94%. We achieved an 80% reduction in human labor using this extraction technique on a data set of 5340 curve sets and expected this efficiency to improve further if applications on future data sets start with this pretrained model. The value of this method is its ability to be applied in bulk and will be more apparent as more data sets are used.

With this new method to classify points for dispersion curve fitting, it is now possible to ingest large volumes of recorded sensor data with minimal human input and then systematically calculate travel times, as shown in Fig. 7. Previously, this data ingestion step for each data set would take hours of human work consisting of point selection by heuristics and experience, but it can now be taken care of with a pipeline designed to separate noise from dispersion curve.

Acknowledgment

The authors would like to thank M. Mousavi and an Anonymous Reviewer for their constructive reviews of this article. They would like to thank Signal Hill Petroleum for permission to use the Long Beach Array and the Southern California Seismic Network for providing data from the broadband stations. They would also like to thank Y. Yue for helpful discussions.

References

[1] N. M. Shapiro, "High-resolution surface-wave tomography from ambient seismic noise," *Science*, vol. 307, no. 5715, pp. 1615–1618, Mar. 2005.

[2] A. L. Levshin, V. Pisarenko, and G. Pogrebinsky, "On a frequency-time analysis of oscillations," in *Annales De Geophysique*, vol. 28, no. 2. Paris, France: Centre National de la Recherche Scientifique, 1972, pp. 211–218.

[3] R. B. Herrmann, "Computer programs in seismology: An evolving tool for instruction and research," *Seismolog. Res. Lett.*, vol. 84, no. 6, pp. 1081–1088, Nov. 2013.

[4] Z. E. Ross, M.-A. Meier, and E. Hauksson, "PWave arrival picking and first-motion polarity determination with deep learning," *J. Geophys. Res. Solid Earth*, vol. 123, no. 6, pp. 5120–5129, Jun. 2018.

[5] W. Zhu and G. C. Beroza, "PhaseNet: A deep-neural-network-based seismic arrival time picking method," *Geophys. J. Int.*, vol. 216, no. 1, pp. 261–273, 2019.

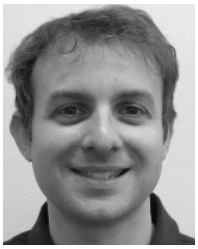
[6] W. Zhu, S. M. Mousavi, and G. C. Beroza, "Seismic signal denoising and decomposition using deep neural networks," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 11, pp. 9476–9488, Nov. 2019.

[7] Z. E. Ross, Y. Yue, M. Meier, E. Hauksson, and T. H. Heaton, "PhaseLink: A deep learning approach to seismic phase association," *J. Geophys. Res. Solid Earth*, vol. 124, no. 1, pp. 856–869, Jan. 2019.
[8] J. Hu, H. Qiu, H. Zhang, and Y. Ben-Zion, "Using deep learning to derive shear-wave velocity models from surface-wave dispersion data," *Seismolog. Res. Lett.*, vol. 91, no. 3, pp. 1738–1751, May 2020.
[9] F.-C. Lin, D. Li, R. W. Clayton, and D. Hollis, "High-resolution 3D shallow crustal structure in Long Beach, California: Application of ambient noise tomography on a dense seismic array," *Geophysics*, vol. 78, no. 4, pp. Q45–Q56, Jul. 2013.
[10] K. Aki and P. Richards, *Quantative Seismology*. Sausalito, CA, USA: Univ. Science Books, 2002.
[11] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
[12] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Proc. Int. Conf. Med. Image Comput. Comput.-Assist. Intervent.* Cham, Switzerland: Springer, 2015, pp. 234–241.
[13] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*. [Online]. Available: http://arxiv.org/abs/1412.6980

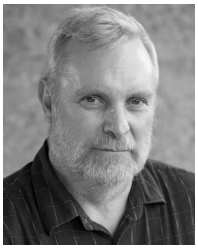
Xiaotian (Jim) Zhang is pursuing the B.S. degree in information and data sciences with Computing and Mathematical Sciences Department, California Institute of Technology (Caltech), Pasadena, CA, USA.

He works in machine learning applications and predictive models with Caltech, where he has performed quantitative research and various trading firms.

Zhe Jia is a Graduate Student in geophysics with Seismological Laboratory, California Institute of Technology (Caltech), Pasadena, CA, USA. His major interests lie in characterizing rupture processes of complex earthquakes and studying the shallow structure with ambient noise tomography.



Zachary E. Ross is an Assistant Professor of geophysics at California Institute of Technology (Caltech), Pasadena, CA, USA, where he uses machine learning and signal processing techniques to better understand earthquakes and fault zones. He is interested in seismicity, earthquake source properties, and fault zone imaging.



Robert W. Clayton is a Professor of geophysics at California Institute of Technology (Caltech), Pasadena, CA, USA, where he works in the areas of seismic wave propagation, earth structure, and tectonics. He has applied imaging methods to the Los Angeles region and to subduction zones around the world.