AGU PUBLICATIONS

Geophysical Research Letters

RESEARCH LETTER

10.1002/2017GL072716

Key Points:

- With trained artificial neural networks, viscoelastic calculations can be accelerated by more than ~55,000%
- Massive viscoelastic calculations, across large spatial and temporal scales and ranges of model parameters, may now take minutes to hours
- These new representations of viscoelastic physics may lead to basic advances in the understanding of the underlying phenomenology

Correspondence to:
P. M. R. DeVries,
phoeberobinson@fas.harvard.edu

Citation:
DeVries, P. M. R., T. B. Thompson, and B. J. Meade (2017), Enabling large-scale viscoelastic calculations via neural network acceleration, *Geophys. Res. Lett.*, *44*, doi:10.1002/2017GL072716.

Received 18 JAN 2017
Accepted 1 MAR 2017
Accepted article online 3 MAR 2017

©2017. American Geophysical Union. All Rights Reserved.

Enabling large-scale viscoelastic calculations via neural network acceleration

Phoebe M. R. DeVries[1], **T. Ben Thompson**[1], **and Brendan J. Meade**[1]

[1]Department of Earth and Planetary Sciences, Harvard University, Cambridge, Massachusetts, USA

Abstract One of the most significant challenges involved in efforts to understand the effects of repeated earthquake cycle activity is the computational costs of large-scale viscoelastic earthquake cycle models. Computationally intensive viscoelastic codes must be evaluated at thousands of times and locations, and as a result, studies tend to adopt a few fixed rheological structures and model geometries and examine the predicted time-dependent deformation over short (<10 years) time periods at a given depth after a large earthquake. Training a deep neural network to learn a computationally efficient representation of viscoelastic solutions, at any time, location, and for a large range of rheological structures, allows these calculations to be done quickly and reliably, with high spatial and temporal resolutions. We demonstrate that this machine learning approach accelerates viscoelastic calculations by more than 50,000%. This magnitude of acceleration will enable the modeling of geometrically complex faults over thousands of earthquake cycles across wider ranges of model parameters and at larger spatial and temporal scales than have been previously possible.

1. Introduction

Viscoelastic coupling across the crust-mantle system is a fundamentally important component of earthquake physics, modulating time-dependent stress transfer, fault loading [e.g., *Roth*, 1988; *Tselentis and Drakopoulos*, 1990; *Freed and Lin*, 2001; *Zeng*, 2001; *Pollitz and Sacks*, 1997, 2002; *Pollitz et al.*, 1998, 2003; *Jónsson et al.*, 2003; *Bürgmann et al.*, 2002], and time-dependent surface deformation [e.g., *Hetland*, 2006; *Hearn et al.*, 2009, 2013]. Computational requirements have often limited the scope of viscoelastic model calculations to a few fixed rheological structures and model geometries over short (<10 years) time periods at a given depth or specific location after a large earthquake [e.g., *Lorenzo-Martín et al.*, 2006; *Pollitz and Sacks*, 2002; *Hearn et al.*, 2009]. The few studies that consider a large suite of viscosity structures [*Takeuchi and Fialko*, 2012; *DeVries et al.*, 2016a] require extensive computation time and resources (in the case of *DeVries et al.* [2016a, 2016b], 2.7 million CPU hours in total). These computational requirements make global-scale viscoelastic calculations across geometrically complex fault systems and large ranges of model parameters prohibitively expensive.

Here we describe how large-scale viscoelastic calculations across geometrically complex fault systems with realistic slip histories can be radically accelerated by training a deep artificial neural network (ANN) to discover computationally efficient representations of a viscoelastic code. Artificial neural networks (ANNs) are machine learning algorithms that are widely used to detect patterns and trends and classify data. The purpose of ANN algorithms for this application can be best understood in the context of two common computational approaches in the Earth Sciences: forward modeling and state estimation methods (Table 1). Consider an arbitrary relationship f between a set of inputs x and a set out outputs y: $y = f(x)$. In simple terms, forward modeling approaches are aimed at solving for the outputs y, given f and inputs x, while state estimation methods exist to solve for a set of inputs x given f and known outputs y. ANN algorithms, in contrast, are not aimed at solving for either inputs or outputs specifically, but rather at finding a new representation of f, the mapping between known inputs x, and known outputs y (Table 1).

Viscoelastic codes take a set of model parameters, for example, the locations of the earthquake source observations points, the source parameters (strike, dip, and rake), viscosity structure, and model geometry, as inputs, and perform computationally intensive calculations in order to ultimately output modeled displacements, velocities, strains, and stresses. Here our goal is to use ANNs to find equivalent and compact representations of the mapping between viscoelastic code inputs and outputs. Once found, these efficient neural network representations may replace computationally intensive viscoelastic codes and allow large-scale viscoelastic calculations to be performed quickly and accurately.

Table 1. Summary of Goals of Three Computational Approaches to Illustrate the Purpose of ANNs for This Application[a]

Computational Approach	Goal
Forward models	Solve for output: $y = f(x)$
State estimation methods	Solve for input: $y = f(x)$
Artificial neural networks (ANNs)	Solve for model: $y = f(x)$

[a]Consider an arbitrary relationship f between a set of inputs x and a set out outputs y: $y = f(x)$. Forward modeling approaches are aimed at solving for the outputs y, given f and inputs x, while state estimation methods exist to solve for a set of inputs x given f and known outputs y. ANN algorithms, in contrast, are not aimed at solving for either inputs or outputs specifically, but rather at finding a new representation of f, the mapping between inputs x, and outputs y.

Below, we first provide an overview of the viscoelastic calculations, followed by an explanation of the structure, development, and implementation of the ANNs used in our application. We then demonstrate the success of these machine learning methods (speedups of >50,000%) for both a simple elastic case and a complex and large-scale viscoelastic case. Finally, we discuss the implications of these methods for earthquake science and modeling efforts across the physical sciences.

2. Viscoelastic Code

Based on the semi-analytic framework developed by *Fukahata and Matsu'ura* [2005, 2006], we have implemented a three-dimensional viscoelastic earthquake cycle model based on a propagator matrix approach that enables the calculation of time-dependent displacements, velocities, strains, and stresses generated in response to fault slip in a coupled crust-mantle system. The distributed architecture of the code (97% efficiency on 1000 cores) facilitates the calculation of viscoelastic effects of earthquakes across realistic fault systems with highly complex geometries and slip histories. However, all told, these calculations may take thousands to millions of CPU-hours for large-scale applications [e.g., *DeVries et al.*, 2016a, 2016b].

3. Artificial Neural Networks (ANNs)

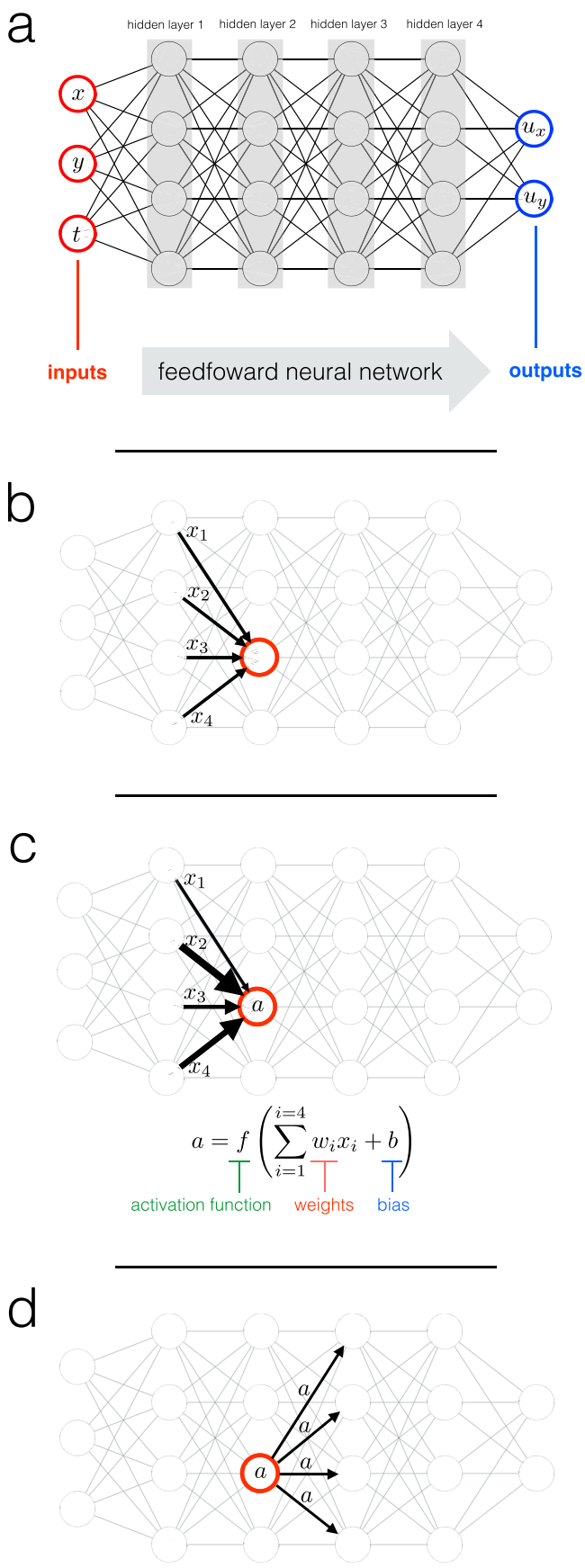
Below, we briefly describe the basic setup, structure, and workflow associated with developing these networks for our specific application; more extensive discussions of ANNs can be found in a number of machine learning textbooks [e.g., *Goodfellow et al.*, 2016; *Neilsen*, 2015].

3.1. Structure

Artificial neural networks consist of many interconnected neurons or individual computational units (Figure 1a) [*Rosenblatt*, 1962]. In feedforward networks, neurons are organized into successive layers (Figure 1a). The first layer (Figure 1a, far left) corresponds to the inputs to the neural network; parameters such as the horizontal (x, y) position of the observations points with respect to the earthquake source; the time after the earthquake; the earthquake source depth, rake, and dip; and the viscosity structure. The last layer of neurons (Figure 1a, far right) corresponds to the predicted outputs of the neural network: displacements in the x, y, and z directions, for example. The interior layers are referred to as "hidden layers" (Figure 1a). More complex ANN structures are used for applications like image classification (convolutional ANNs [e.g., *LeCun et al.*, 1998a]) and speech recognition (recurrent ANNs [e.g., *Graves et al.*, 2013]), but for our application, feedforward networks perform extremely well.

Within ANNs, each neuron performs a simple operation: for input values x_i from the previous layer (Figure 1b), each neuron outputs a single value $a = f(\sum w_i x_i + b)$ (Figures 1c and 1d). Activation functions f are application-dependent [*LeCun et al.*, 1998b; *Glorot and Bengio*, 2010] but are usually chosen to be nonlinear or piecewise functions (e.g., the sigmoid or hyperbolic tangent functions; Figure 1c). The values of the weights w_i and the bias b are unique to each individual neuron (Figure 1c); these are the parameters that are iteratively adjusted during the training of the neural network. Neuron j in layer l accepts the outputs $a^{l-1}_{1\ldots N}$ from all N neurons in the previous layer $l-1$ as its inputs, and outputs a single value $a^l_j = f\left(\sum w_i a^{l-1}_i + b\right)$ that is sent to every neuron in layer $l+1$ (Figures 1b–1d) [*Neilsen*, 2015].

In practice, despite the universality principle for simple nets [e.g., *Cybenko*, 1989; *Hornik et al.*, 1989], researchers tend to choose the deepest (number of hidden layers) nets possible within their computational constraints.



3.2. Training and Testing

The workflow for developing ANNs consists of two phases: (1) training and validation and (2) testing. Both phases require inputs, in our case, sample inputs to the viscoelastic code, and associated code outputs. We build the training, validation, and testing data sets by running the viscoelastic code across a range of model parameters. All the data sets consist of vectors of input parameters (e.g., *x* and *y* locations relative to the horizontal position of the source) and output values from the viscoelastic code (e.g., the *x*, *y*, and *z* displacements associated with each set of input values). As described above, the neural network uses the input parameters as the first layer of neurons, and the last layer of neurons represents the predicted output values (Figure 1a).

During the training and validation phase, a cost (or loss) function C, the mean squared error, in this case, is used to evaluate the performance of the ANN by comparing the viscoelastic code outputs (from the training and validation data sets) to the values predicted by the ANN. Optimization methods like stochastic gradient descent are used to iteratively solve for the different weights w_i and biases *b* for each neuron that allow the ANN to best approximate the training data outputs. At each iteration, a backpropagation algorithm estimates the partial derivatives of the cost

Figure 1. (a) Structure and setup of an example artificial neural network with hidden layers, input layers, and output layers highlighted. Calculations move from left to right in the feedforward neural networks used here. (b) Inputs to an example neuron from the previous layer. (c) The calculation performed by the example neuron, weighting the inputs relative to one another, adding a bias, and applying an activation function, in order to calculation a value *a*, referred to as the activation of the neuron. (d) The activation of the example neuron serves as one of the inputs to the next layer of neurons. Each neuron in the successive layers of the ANN are performing this same operation with different values of the tunable parameters *b* and w_i.

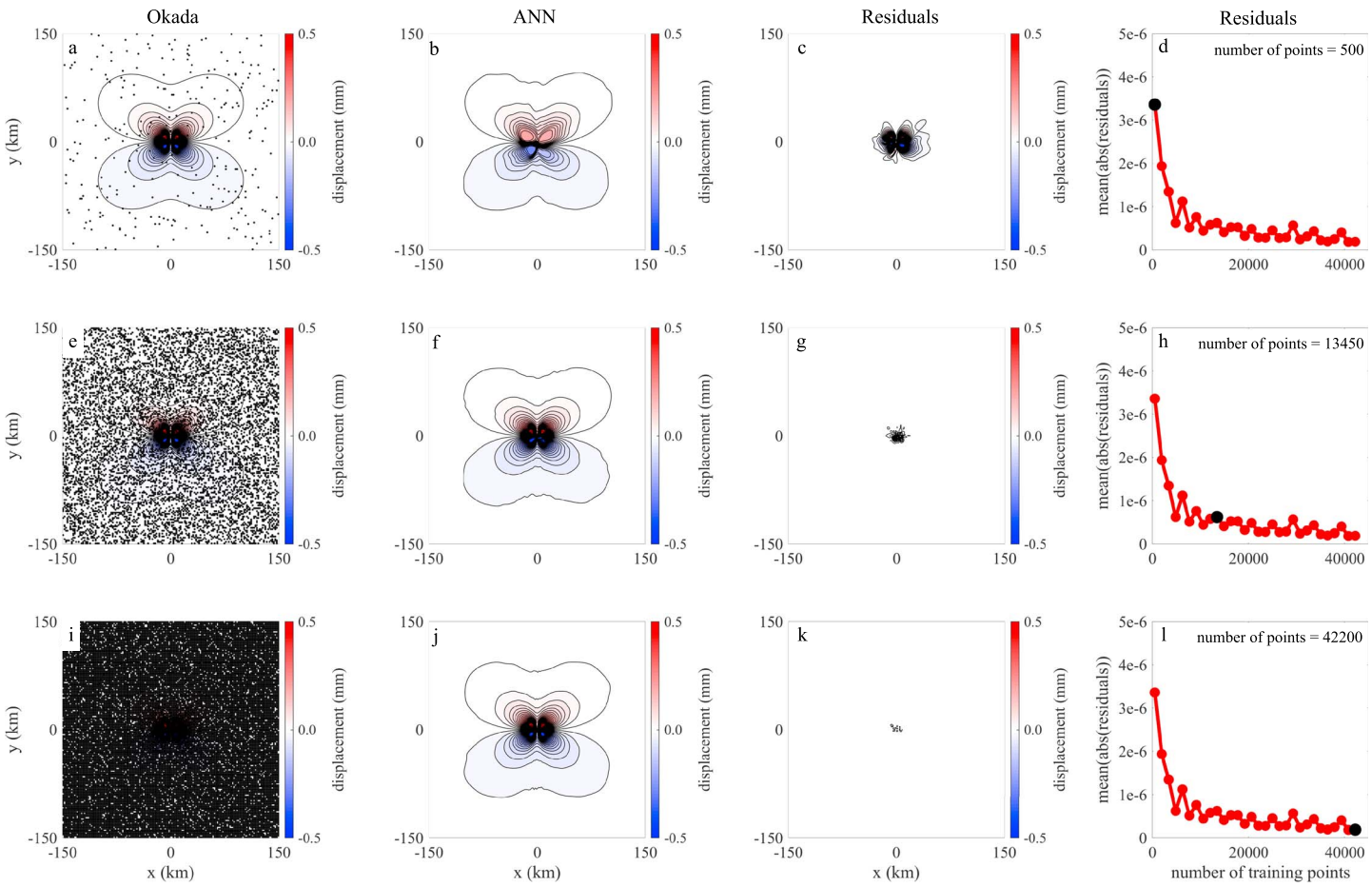


Figure 2. Performance of the artificial neural network for a simple elastic case. (a) *Okada* [1992] solution for a point source at 7.9 km depth. The black dots indicate 500 randomly distributed training points used to train an ANN. (b) Predictions from this ANN. (c) Residuals between the ANN prediction and the true *Okada* [1992] solution. (d) Mean absolute residuals as a function of the number of training points for this elastic case. (e–h) Analogous to Figures 2a–2d for 13,450 training points. (i–l) Analogous to Figures 2a–2d for 42,200 training points.

function with respect to incremental changes of all the weights w_i and biases b, to determine the gradient descent directions for the next iteration.

Validation data are not used in the optimization or backpropagation algorithms. Instead, the cost function evaluated over the validation data serves as an independent metric of the performance of the ANN; a validation data cost function that is increasing with each iteration would suggest that the net is overfitting, or memorizing, the training data, at the cost of generalizability to other inputs. Dealing with overfitting is not as significant a challenge for us as it is in other ANN applications, because there is no noise in the training data set in this case.

Finally, in the testing phase, we test the predictions of the ANN against the results from the viscoelastic code across different parameter values than the training set. This testing phase is necessary to evaluate how accurately the ANN can approximate the code outputs in a more general sense.

3.3. Implementation

We use the Python toolkit Keras (https://github.com/fchollet/keras), which provides a high-level application programming interface to access the Theano (http://deeplearning.net/software/theano/) and TensorFlow (https://www.tensorflow.org/) deep learning libraries. For the examples shown here, we use Theano and an adaptive moment estimation (Adam) optimization method, a stochastic gradient descent algorithm that computes adaptive learning rates [*Kingma and Ba*, 2015].

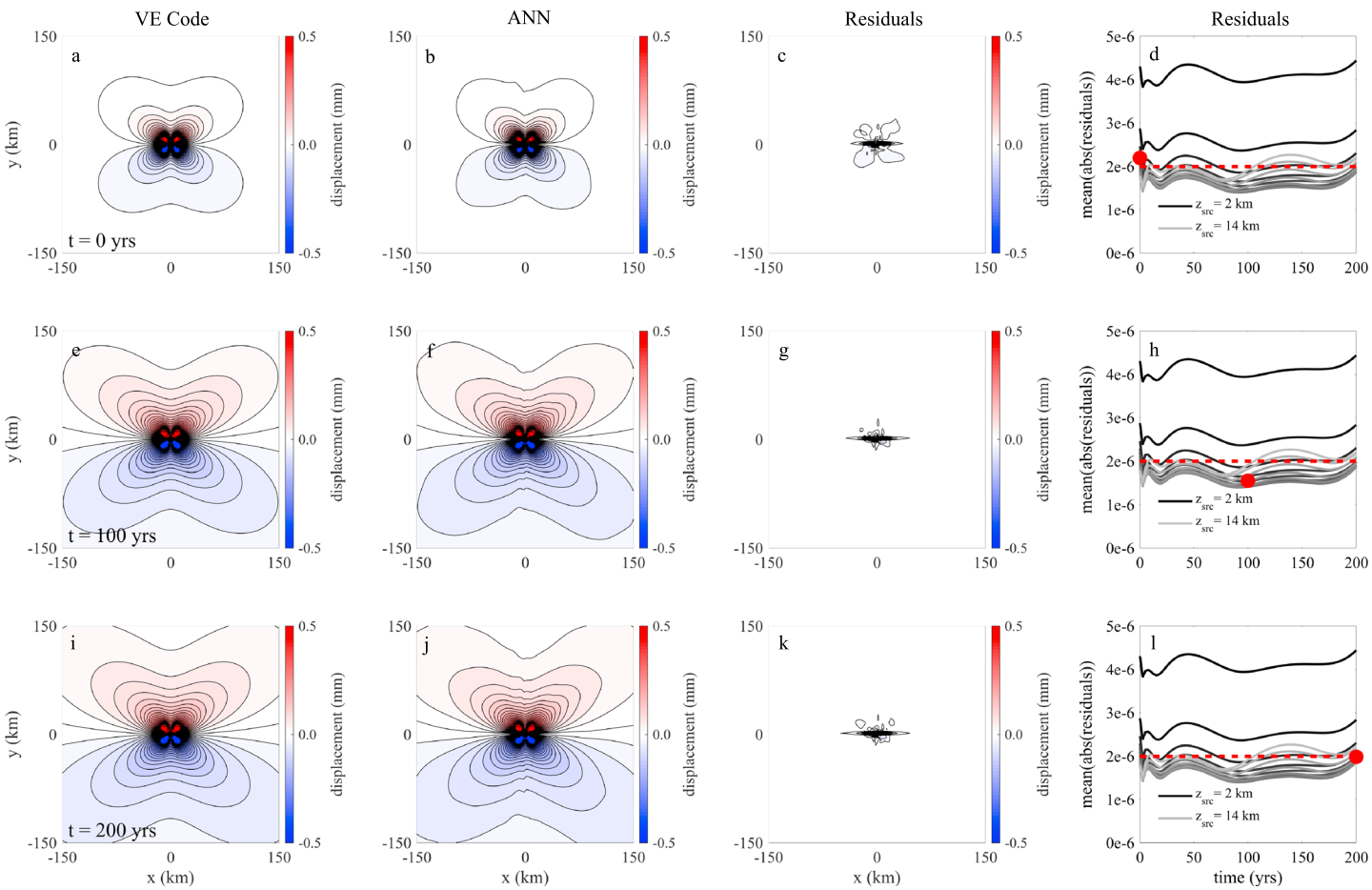


Figure 3. Performance of the artificial neural network for a complex and large-scale viscoelastic case. (a) Viscoelastic code solution for a point source at 6 km depth for a Maxwell rheology of 10^{19} Pa s at $t = 0$ year after an earthquake. (b) ANN prediction for comparison. (c) Residuals between the true viscoelastic solution and the ANN prediction. (d) Mean absolute residuals as a function of time, averaged across all viscosity structures tested, for earthquake source depths between 2 and 14 km. (e–h) Analogous to Figures 3a–3d for $t = 100$ years after an earthquake. (i–l) Analogous to Figures 3a–3d for $t = 200$ years after an earthquake.

4. Results

Below we demonstrate the success of these methods, in terms of both accuracy and acceleration relative to existing codes, for a simple elastic case [*Okada*, 1992] and for a complex and large-scale viscoelastic case.

4.1. Elastic Case

We consider the simple case first: the displacements at the surface due to a strike-slip point source in an elastic half-space at a depth of 7.9 km [*Okada*, 1992]. In this case, the ANN is essentially acting as a two-dimensional interpolator: the inputs are *x* and *y* locations, and the outputs are *x*, *y*, and *z* displacements. A neural network trained on only 500 randomly distributed data points does not perform particularly well (mean residual of ~3.4×10^{-6} mm over a grid of 32,580 test points; Figure 2); however, as the density of training data is increased to over 40,000 points, the accuracy of the ANN solution rapidly improves (a mean residual of ~1.9×10^{-7} mm over a grid of 32,580 test points; Figure 2). The maximum absolute value of the *x* displacements is ~0.5 mm, suggesting that this best performing ANN is accurate to within two parts in 5 million on average. In addition, the ANN performs these calculations 165% faster than the explicit *Okada* [1992] calculation.

4.2. Viscoelastic Case

This method also performs extremely well for a large-scale viscoelastic case involving six model parameters (Figure 3). In this more complex case, we fix the model geometry to be a 15 km thick elastic upper crust

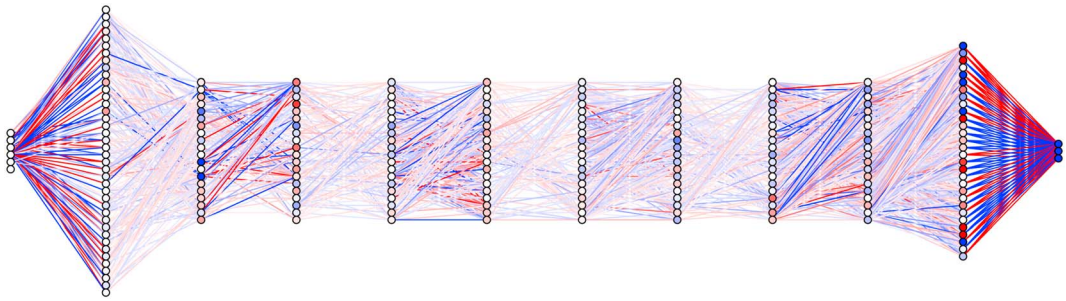


Figure 4. Visualization of the structure, weights, and biases of one of the most successful trained ANNs for the complex viscoelastic case (Figure 3). The circles represent individual neurons (as in Figure 1); the colors of the connections between neurons encode the magnitude and sign (red positive, blue negative) of each individual weight, while the colors of the neurons represent the sign and magnitude of the bias associated with each neuron. The information contained in this visualization (ANN structure, weights, and biases) is, in some sense, a new and compact representation of viscoelastic physics.

overlying a viscoelastic half-space with a transient Burgers rheology (shear moduli $\mu = 30$ GPa in both layers) and consider only strike-slip sources. The input parameters to the artificial neural network are the (x, y) locations of the observations points at the surface relative to the source location, the depth of the earthquake source, the time after the earthquake, and the Maxwell and Kelvin viscosities η_M and η_K that characterize a Burgers rheology. The outputs predicted by the neural network are, as in the elastic case, x, y, and z displacements.

For this large-scale test, the training data consist of 236,792,880 sets of inputs and outputs (~17 GB of training data in total): specifically, 10,609 (x, y) observation points at the surface, 20 source depths, 31 times, and 36 viscosity structures as inputs, and three components of displacement as outputs. In contrast to many ANN applications (e.g., voice recognition and image classification), we can exactly choose the training data in order to try to optimize the training of the ANNs. Training points are not regularly spaced in time, for example, because best results were obtained when the viscoelastic code solutions were sampled according to a quadratic sequence in time, in order to feed more information about the rapidly changing solution near $t = 0$ into the ANN.

We tested a range of deep ANN structures (10–40 neurons per layer and 2–10 hidden layers) and a number of different activation functions (hyperbolic tangent, sigmoid, and rectified linear units (ReLU), the last defined as $f(x) = x$ for $x > 0$ and $f(x) = 0$ for $x < 0$). The most successful nets (mean absolute residual of ~2×10^{-6} mm; Figure 3) incorporate alternating layers of hyperbolic tangent functions and ReLUs, with at least 10 neurons per layer and at least six hidden layers (e.g., Figure 4).

The performance of these ANNs, once trained, was evaluated comprehensively against the true viscoelastic solutions at 10,000 (x, y) locations for each of 13 source depths, 25 viscosity structures, and 81 times (Figure 3). The mean absolute residual, across all of these parameters, was ~2×10^{-6} mm; the performance improves for deeper source depths but has no clear pattern as a function of time (Figures 3d, 3h, and 3l). Perhaps, most significantly, for this large-scale test (Figure 3), the viscoelastic code took 419,790 s to generate the true solution. The ANN prediction took 6366 s on a single CPU, a speedup of 6600% compared to the viscoelastic code. Running the ANN prediction on a GPU further accelerates the same calculation to 763 s, a speedup of 55,018% over the viscoelastic code.

5. Discussion and Implications

Artificial neural networks can be used to discover new, compact, and accurate computational representations of viscoelastic physics (e.g., Figure 4). In other words, the structure, weights, and biases of trained ANNs are an alternative way to express the complex physics within existing computationally intensive viscoelastic codes. These new and efficient computational representations of viscoelastic physics are notable in a number of ways, listed below:

1. *Acceleration*: With trained ANNs, viscoelastic calculations can be accelerated by ~55,000%. Large-scale viscoelastic calculations [e.g., *DeVries et al.*, 2016a, 2016b] that previously took over 2 million CPU hours could take only ~5 h on an equivalent number of GPUs. With these trained ANNs, massive viscoelastic

calculations, across very large spatial and temporal scales and ranges of model parameters, can be done in a matter of minutes or hours.

2. *Ease of use and portability*: To run forward predictions using the trained ANNs, only the matrices of weights w_{ij}, biases b_j, and activation functions are required. This information can generally be contained in a small (<50 KB) data file. Therefore, with only an ~100 line Python or MATLAB script and a small data file, time-dependent viscoelastic deformation at any observation point and time, due to any source location, fault geometry, and rheological model, will be both easy and fast to calculate on any device with an installation of MATLAB or Python.
3. *Unique physical insights*: From a philosophical perspective, these new ANN representations of viscoelastic physics may lead to basic advances in the understanding of the underlying phenomenology and physics. For example, a visualization of one successful neural network (Figure 4) reveals that the weights connecting viscosity inputs (Maxwell and Kelvin viscosities η_M and η_K) to the first hidden layer are on average at least a factor of 10 smaller in magnitude than all other connections. Therefore, in some sense, the viscosity structure in a viscoelastic calculation is an order of magnitude less influential over the full solution than other model parameters, such as the location of the source and observations points. Artificial neural networks have to the potential to illuminate countless fundamental physical insights along these lines.
4. *Implications for modeling across the physical sciences*: More broadly, this ANN method is exciting because it could be applied to virtually any type of physical model, accelerating model calculations and enabling larger parameter spaces and spatial and temporal scales to be explored in detail. These methods could therefore lead to novel insights across the physical sciences.

6. Conclusions

We enable viscoelastic calculations on a global scale across geometrically complex fault systems by training deep neural networks to accurately represent viscoelastic solutions. This method leads to accurate results with speedups of at least ~55,000%, allowing massive viscoelastic calculations to be done in minutes to hours, compared to the millions of CPU hours required to run large-scale viscoelastic calculations with existing codes. Once trained, artificial neural networks are new and compact representations of viscoelastic calculations, and as such may provide basic insights into viscoelastic physics and phenomenology. More broadly, this artificial neural network approach could be applied to virtually any physical model and contribute to the understanding of complicated physical behavior across many scientific fields.

Acknowledgments
We thank Eric Dunham, two anonymous reviewers, and Editor Andrew Newman for thoughtful reviews. No data was used in producing the results in this manuscript. All data referred to in the discussion are properly cited and referred to in the reference list. The computations in this paper were run on the Odyssey cluster supported by the FAS Division of Science, Research Computing Group at Harvard University. This work was supported by the Harvard University, the Southern California Earthquake Center (contribution 6239) funded by NSF Cooperative Agreement EAR-1033462 and USGS Cooperative Agreement G12AC20038, and the Department of Energy Computational Science Graduate Fellowship Program of the Office of Science and National Nuclear Security Administration in the Department of Energy under contract DE-FG02-97ER25308.

References

Bürgmann, R., S. Ergintav, P. Segall, E. Hearn, S. McClusky, R. E. Reilinger, H. Woith, and J. Zschau (2002), Time-dependent distributed afterslip on and deep below the Izmit Earthquake rupture, *Bull. Seismol. Soc. Am.*, *92*(1), 126–137.

Cybenko, G. (1989), Approximation by superpositions of a sigmoidal function, *Math. Control, Signals Syst.*, *2*, 303–314.

DeVries, P., P. Krastev, J. Dolan, and B. Meade (2016a), Viscoelastic block models of the North Anatolian fault: A unified earthquake cycle representation of pre- and post-seismic observations, *Bull. Seismol. Soc. Am.*, *107*, doi:10.1785/0120160059.

DeVries, P., P. Krastev, and B. Meade (2016b), Geodetically constrained stress transfer and earthquake triggering along the North Anatolian fault, *Geochem. Geophys. Geosyst.*, *17*, 2700–2716, doi:10.1002/2016GC006313.

Freed, A. M., and J. Lin (2001), Delayed triggering of the 1999 Hector Mine earthquake by viscoelastic stress transfer, *Nature*, *411*(6834), 180–183.

Fukahata, Y., and M. Matsu'ura (2005), General expressions for internal deformation fields due to a dislocation source in a multilayered elastic half-space, *Geophys. J. Int.*, *161*(2), 507–521.

Fukahata, Y., and M. Matsu'ura (2006), Quasi-static internal deformation due to a dislocation source in a multilayered elastic/viscoelastic half-space and an equivalent theorem, *Geophys. J. Int.*, *166*(1), 418–432.

Glorot, X., and Y. Bengio (2010), Understanding the difficulty of training deep feedforward neural networks, in *Proceedings of AISTATS*, vol. 9, pp. 249–256, Chia Laguna Resort, Sardinia, Italy.

Goodfellow, I., Y. Bengio, A. Courville (2016), Deep learning, MIT Press. [Available at http://www.deeplearningbook.org.]

Graves, A., A. Mohamed, and G. Hinton (2013), Speech recognition with deep recurrent neural networks, 2013 IEEE international conference on acoustics, speech and signal processing, pp. 6645–6649, Retrieved November 20, 2016, from the arXiv database (arXiv: 1303.5778v1).

Hearn, E. H., S. McClusky, S. Ergintav, and R. E. Reilinger (2009), Izmit earthquake postseismic deformation and dynamics of the North Anatolian Fault Zone, *J. Geophys. Res.*, *114*, B08405, doi:10.1029/2008JB006026.

Hearn, E. H., C. T. Onishi, F. F. Pollitz, and W. R. Thatcher (2013), How do "ghost transients" from past earthquakes affect GPS slip rate estimates on southern California faults?, *Geochem. Geophys. Geosyst.*, *14*, 828–838, doi:10.1002/ggge.20080.

Hetland, E. (2006), Models of interseismic deformation with an analytic framework for the inclusion of general linear viscoelastic rheologies, PhD thesis, Mass. Inst. of Technology, 255 p.

Hornik, K., M. Stinchcombe, and H. White (1989), Multilayer feedforward networks are universal approximators, *Neural Networks*, *2*(5), 359–366.

Jónsson, S., P. Segall, R. Pedersen, and G. Bjornsson (2003), Post-earthquake ground movements correlated to pore-pressure transients, *Lett. Nat.*, *424*, 179–183.

Kingma, D., and J. Ba (2015), Adam: A method for stochastic optimization, Conference paper at the 3rd International Conference for Learning Representations, San Diego, Retrieved October 25, 2016, from the arXiv database (arXiv: 1412.6980v8).

LeCun, Y., L. Bottou, Y. Bengio, and P. Haffner (1998a), Gradient-based learning applied to document recognition, *Proc. IEEE*, *86*(11), 2278–2324.

LeCun, Y., L. Bottou, G. Orr, and K.-R. Müller (1998b), Efficient backprop, in *Neural Networks: Tricks of the Trade* vol. 1524, pp. 9–50, Springer, Berlin.

Lorenzo-Martín, F., F. Roth, and R. Wang (2006), Elastic and inelastic triggering of earthquakes in the North Anatolian Fault zone, *Tectonophysics*, *424*(3), 271–289.

Neilsen, M. A. (2015), *Neural Networks and Deep Learning*, Determination Press. [Available at http://neuralnetworksanddeeplearning.com.]

Okada, Y. (1992), Internal deformation due to shear and tensile faults in a half-space, *Bull. Seismol. Soc. Am.*, *82*(2), 1018–1040.

Pollitz, F. F., and I. S. Sacks (1997), The 1995 Kobe, Japan, earthquake: A long-delayed aftershock of the offshore 1944 Tonankai and 1946 Nankaido earthquakes, *Bull. Seismol. Soc. Am.*, *87*(1), 1–10.

Pollitz, F. F., and I. S. Sacks (2002), Stress triggering of the 1999 Hector Mine earthquake by transient deformation following the 1992 Landers earthquake, *Bull. Seismol. Soc. Am.*, *92*(4), 1487–1496.

Pollitz, F. F., R. Bürgmann, and B. Romanowicz (1998), Viscosity of oceanic asthenosphere inferred from remote triggering of earthquakes, *Science*, *280*(5367), 1245–1249.

Pollitz, F., M. Vergnolle and E. Calais (2003), Fault interaction and stress triggering of twentieth century earthquakes in Mongolia, *J. Geophys. Res.*, *108*(B10), 2503, doi:10.1029/2002JB002375.

Rosenblatt, F. (1962), *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*, pp. 616, Spartan Books, Washington, D. C.

Roth, F. (1988), Modelling of stress patterns along the western part of the North Anatolian fault zone, *Tectonophysics*, *152*(3), 215–226.

Takeuchi, C., and Y. Fialko (2012), Dynamic models of interseismic deformation and stress transfer from plate motion to continental transform faults, *J. Geophys. Res.*, *117*, B05403, doi:10.1029/2011JB009056.

Tselentis, G. A., and J. Drakopoulos (1990), Stress transfer and nonlinear stress accumulation at the North Anatolian fault, Turkey, *Pure Appl. Geophys.*, *132*(4), 699–710.

Zeng, Y. (2001), Viscoelastic stress-triggering of the 1999 Hector Mine Earthquake by the 1992 Landers Earthquake, *Geophys. Res. Lett.*, *28*, 3007–3010, doi:10.1029/2000GL012806.