# Tutorial: a geophysical perspective on machine learning

Ian F. Jones[1*]

**Abstract**

Machine learning is fast becoming omnipresent in all aspects of science involving large volumes of data. The terminology and ideas involved are not immediately transparent or obvious to people not already immersed in the minutia of the implementation of machine learning procedures. Here, I try to relate the concepts and terminology used in machine learning to things that geoscientists will already be familiar with, so as to form a bridge between their current knowledge and an understanding of machine learning.

## Introduction

I first attended a talk on what was then billed as artificial intelligence, specifically the application of neural networks in geophysics, over 20 year ago. I understood nothing. The terminology was totally alien, and I had no conceptual links to relate what the speaker was saying to what I already knew. In terms of the science, he was essentially speaking a 'foreign language'.

The application of artificial intelligence, deep learning, and machine learning (ML) have since become prevalent, and are fast becoming omnipresent in all aspects of science that involve handling large data volumes of any sort.

In order to try to gain a better grasp of the underlying principle of machine learning, I'll try to relate the ideas and terminology used in ML to concepts and terms that will already be familiar to geophysicists with a basic understanding of common geophysical topics such as deghosting (a formal description of methods like deconvolution or inversion in the context of ML can be found, for example in: Calderón-Macías et al., 1997; Russell, 2019).

## An analogy to de-ghosting procedures

Consider a single seismic trace from a marine survey, and assume that we know that the underlying source generates a band-limited minimum phase wavelet characteristic of an airgun array. This minimum phase signal then becomes modified by the source and receiver ghosts (reflections of the sea surface) to produce the source signature present in the recorded data. These ghosts are delayed opposite polarity versions of the initial source, and knowledge of the source and receiver depths make them predictable in terms of the notches in the spectrum

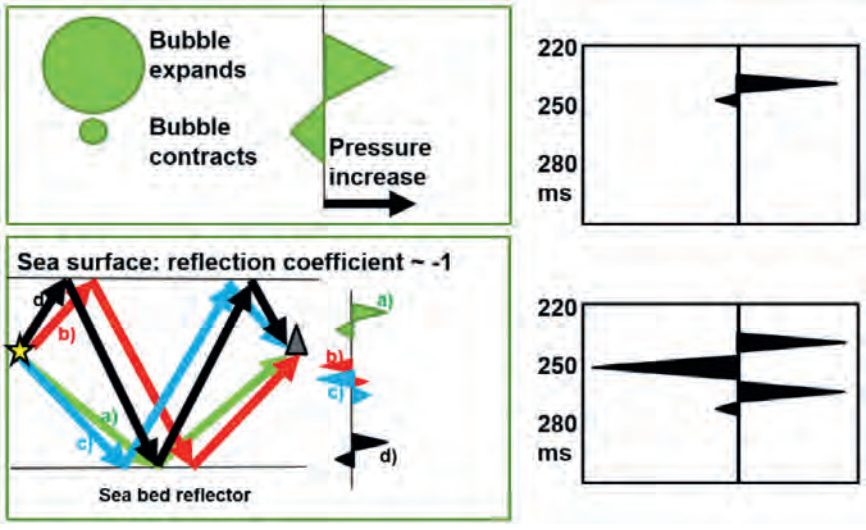


**Figure 1** Left top: the bubble expansion from firing the airgun and subsequent bubble contraction forms a minimum phase pulse; bottom: for a source (yellow star) and receiver (grey triangle) below the sea surface, we have a) the primary reflection (green), b) the source-side ghost (red), c) the receiver-side ghost (blue) and d) the double surface bounce ghost (black). As the receiver is usually deeper than the source, the receiver ghost arrives slightly after the source ghost, but the two pulses usually combine to form a large reverse-polarity peak, which arrives at the water-depth travel time (as the increase in travel time due to the ghost travel path compensates for the reduction in travel time due to the source and receiver being below the sea surface). Right: for a source at 7.5-m depth (10ms) and receiver at 9 m (12ms), an input waveform (top) is modified to create the typical marine wavelet (bottom).

[1] ION
[*] Corresponding author, E-mail: Ian.Jones@iongeo.com
DOI: 10.3997/1365-2397.fb2021017

of the source signature. However, this delay varies with angle of incidence of the raypaths, i.e. with offset between source and receiver. Figure 1 shows a cartoon of this process: as well as the sea surface ghost reflections, there is also the effect of the contraction of the bubble pulse (and later delayed repetitions of the oscillating bubble). In simple convolutional theory, we assert that the measured (observed) trace is composed of the Earth's (approximately white) reflectivity series, convolved with the band-limited minimum phase source wavelet, the source-side and receiver-side ghost responses, the receiver instrument response, and finally then all damped with a time and frequency varying amplitude decay factor (made up of a divergence component and various attenuative absorption components), plus noise.

In contemporary data processing and imaging workflows, we often desire to suppress the source and receiver ghosts so as to obtain data with a broader bandwidth and thereby provide higher-resolution images (Zhou et al., 2012). We typically address this problem by designing and applying a deghosting operator (and the nature of this procedure can vary depending on the water depth: if the water is shallow then the direct wave obscures the seabed reflection events and restricts use of some methods).

The deghosting operator design could perhaps involve a 3D transform and migration-like operation to map all related notches to the same spectral location in order to minimize the notch dependency on angle of incidence Then for specified source and receiver depths and sea-surface reflection coefficients, a deghosting operator would be designed to try to minimize the effect of the notches in the amplitude and phase spectra. The operators would be specified with certain design parameters, and the field data may be preconditioned with some form of amplitude decay compensation and noise suppression. The operators may also be smoothed laterally so as to promote stability in the face of rapidly varying sea states.

Now consider trying to automate this approach for any marine seismic data. First, take a modelled signature for an airgun array at source and receiver depths representative of this survey without any ghosts included (the desired output), and, after suitable normalization and alignment, subtract this from the data samples of the deghosted field data wavelet (the actual output). Next, calculate the sum-of-squares of the sample-by-sample differences to create a single number characterizing the overall difference between the desired and actual output (and call this number the 'cost function'). In the ideal case the cost function will be small, as the deghosted wavelet will closely resemble the modelled signature.

The estimate could be refined by setting-up a least-squares loop to minimize the cost function by adjusting the parameters of the deghosting procedure (e.g. the reflection coefficients, true source and receiver depths, operator length, etc.). There are perhaps a dozen parameters in the deghosting workflow and consequently, in routine production testing, it is totally impractical to manually investigate the interplay of all possible combinations of these parameters even for a limited range of their values. However, in a least-squares multivariate analysis procedure this is a relatively straightforward task (albeit compute intensive).

Such parameter optimization constitutes the first step in a machine learning procedure.

After several iterations of such least-squares minimization, we should have an optimal set of parameters for the deghosting procedure to correct for the effect of the ghosts on these data. Furthermore, we could extend the procedure to use input traces from a suite of similar surveys, all of the same known airgun configuration, source and receiver depth, thus having the same underlying wavelet shape, and thereby obtain a more optimal set of parameters. These parameters will now be more independent of the undesired influence of slight variations in source or receiver depths, and swell and other noise, or of the Earth's reflectivity. The cost function will now be very small, as the deghosted wavelet extracted from field data with the optimized parameters should very closely resemble the modelled library signature. Least squares minimization essentially back-propagates the observed error (as measured by the cost function) through the various steps in the processing sequence, by asking: what parameter changes in each of the preceding steps will result in a reduced cost function?

These steps of automated parameter optimization can now be extended into a '*learning*' phase to more fully automate and generalize the deghosting procedure for data with *any* source and receiver depths, by proceeding as follows. Take field data from a wide range of surveys and group them into classes for a given combination of say 40 source and 40 receiver depths, for tow depths of, for example, between 3 m to 22.5 m in steps of 0.5 m (thus giving 1600 groups), and for each of these 1600 groups create a modelled library signature. Next repeat the least-squares parameter optimization procedure for each of the 1600 groups of source and receiver depth combinations. In terms of ML terminology, this extensive task of parameter optimization for each of the 1600 groups is referred to as '*training*' and the underlying data used in this process as the '*training set*'. For whatever deghosting procedure we use, we now have a library of least-squares optimized parameters for data acquired with many source and receiver depths.

Least-squares cost-function minimization is a widely used technique for parameter optimization, and further details on this item can be found for example in Shewchuck (1994), Schleicher (2018), and Jones et al., (2019). It can be noted that the task we undertake (from the perspective of the maths) constitutes attempting to find the inverse of a large matrix system (the Hessian). Consequently, the heart of a ML system is the use of linear algebra to address a highly non-linear problem in an approximate linearized iterative way.

## Using the optimized parameters

Now turn the problem around: given some new and hitherto unseen marine seismic data, we want to determine which deghosting filter is most appropriate for it by comparing it to the 1600 groups that we previously analysed. Rather than trying to extract a deghosted broadband wavelet from the new data by designing a bespoke optimized processing scheme, instead we simply apply the aforementioned deghosting procedure 1600 times with each of the previously determined optimized parameters for each of the 1600 groups. This is a very quick procedure, as we already

have all of the processing parameters from the '*training*' steps. We can then compute the cost function for all 1600 of these output deghosted wavelets (i.e. sum-squares difference between the 1600 modelled signatures and the deghosted wavelet estimate output for the new data with each of the 1600 parameter sets). Finally, we classify the new data by saying that it is most likely to belong to the source and receiver depth group that gave the smallest cost-function. This final step is the classification of the output into the desired descriptive classes.

It should be noted that these classification results are not absolute, nor guaranteed to be correct; it is perhaps better to think of the cost function as representing the *probability* that the new trace belongs to a certain class. If we input a trace of data from the soundtrack off a music album, the classification would still give us an answer in terms of the likely source and receiver depth class and corresponding deghosting filter, but this time the answer would clearly be meaningless. Or, if we provided marine airgun data with a receiver depth of 30 m (i.e. outside the range of the training), it would incorrectly attribute this new trace to having a probable receiver depth between 3 and 22.5 m. However, for both these examples of misclassification, the cost functions would be relatively large (i.e. the *probability* of belonging to one of our 1600 classes would be small).

## Terminology

As noted earlier, the terminology used in ML can be opaque, so in this section I'll outline some of the common terms employed in the ML literature and relate them to terminology more familiar to geoscientists.

Each individual input sample from the seismic data being inputted into the system is referred to as a '*neurone*', as it can influence the outcome (Rumelhart et al., 1986). The processing sequence designed to estimate the underlying deghosting operator and effectively remove the ghost response is referred to as the '*network*'. Each step within that network is referred to as a '*layer*'. If we packaged this processing sequence as a standalone 'black box' workflow, we could think of its inner working as comprising '*hidden layers*'. Some of those internal processes (*layers*) were designed to reduce the number of data samples whilst retaining information content (e.g. smoothing and decimation); such steps are referred to as '*down-sampling*'. Some steps, such as scaling or normalization, could be designed to emphasize specific aspects of the data, and these are referred to as '*activation functions*', a common one being to take only positive values ('*rectify*') and then to normalize the data with, say, a linear ramp (known as *ReLU*: rectified linear unit), or perhaps a sigmoidal function. As noted earlier, the procedure of iteratively minimizing the cost-function to optimize all the processing parameters in the network is referred to as the '*training*' of the network. A more extensive list of the terms commonly used in ML can be found, for example, on the Wikipedia page cited in the references.

As can be imagined, the key to getting a useful and usable neural network, is to have an understanding of what individual processing steps (the '*hidden layers*') will best help us to achieve our objectives. Then for very large datasets we 'simply' need an enormous computer to estimate the inverse of the Hessian for each of the classes we train for (although various approximate solutions can be found more cost-effectively).

The training itself will generally be extremely user intensive as someone has to select, '*label*', and input the training data. However, this procedure can be streamlined by using neural networks (NN) themselves to generate additional input datasets (and if performed without user intervention, this is said to be '*unsupervised*'). Each of these computed (synthetic) training datasets would be constructed so as to have similar statistical behaviour to the real data sets. Then, using pairs of neural networks, each would feed the other one of its created synthetic datasets (in addition to the original real datasets), giving rise to an '*augmented*' NN that might be of use when working with limited numbers of input field data sets. In the case where one network '*generates*' the input and one network '*discriminates*' against poor solutions based on the cost function, this approach constitutes an '*adversarial*' competition with one NN testing the other, and is referred to as a '*generative adversarial network*' (GAN).

An inherent limitation for any ML procedure is the degree of generality of the training procedure: can the training be sufficiently universally diagnostic, such that *any* new example can be correctly classified? It would be of little use if we have to retrain the network for every single problem that we encountered.

## ML for other geophysical tasks.

Another view of the procedure is to think of ML as trying to identify the most important and diagnostic aspects of a data-set, and then boiling these down into the smallest possible set of descriptors that adequately capture the vast majority of the information contained in the data (in terms of the mathematics, this is searching for the optimal and most condensed basis functions to represent the data). These descriptors can then be used to assess if a given new data-set resembles some target class, and thereby categorize a new hitherto unseen object as belonging or not belonging to a given class.

Let us say that we want to train a NN to interpolate missing traces from shot gathers: take a shot gather and replace several traces with blank (zero-value) traces. Now run the NN to minimize the misfit between the original and decimated gathers, and do this for thousands of input gathers with very many permutations of zeroed-out traces. At the end of this training procedure, the NN will have determined an optimum way of filling-in the gaps between the live traces. This NN can then be applied to new, hitherto unseen shot gathers so as to interpolate (reconstruct) missing traces (with some limiting assumptions, such as perhaps the maximum offset, trace length, and sample rate being the same, etc.). With a more conventional approach, this interpolation/reconstruction might have been performed using a parabolic Radon transform following normal-moveout correction (and in this case, the basis functions would have been the Radon p-traces). A similar approach can be taken to extrapolate near-traces back to zero offset to facilitate the workings of SRME (Verschuur, 2020). It is important and instructive to note that when we perform a parabolic Radon transform to decompose a moveout corrected shot gather; we are basing this choice of basis vectors on the

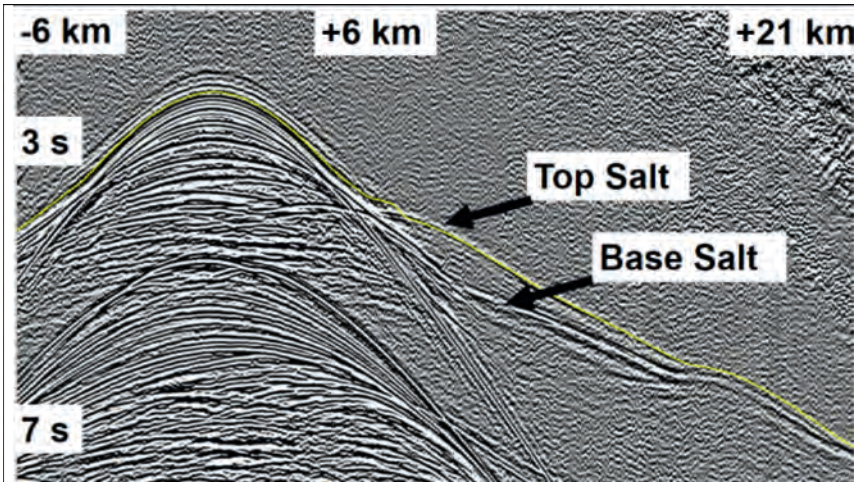


**Figure 2** A common receiver gather from long offset ocean bottom node data. The yellow line indicates the ML picks of the first arrivals for use in waveform inversion (courtesy of Yannick Cobo, ION).

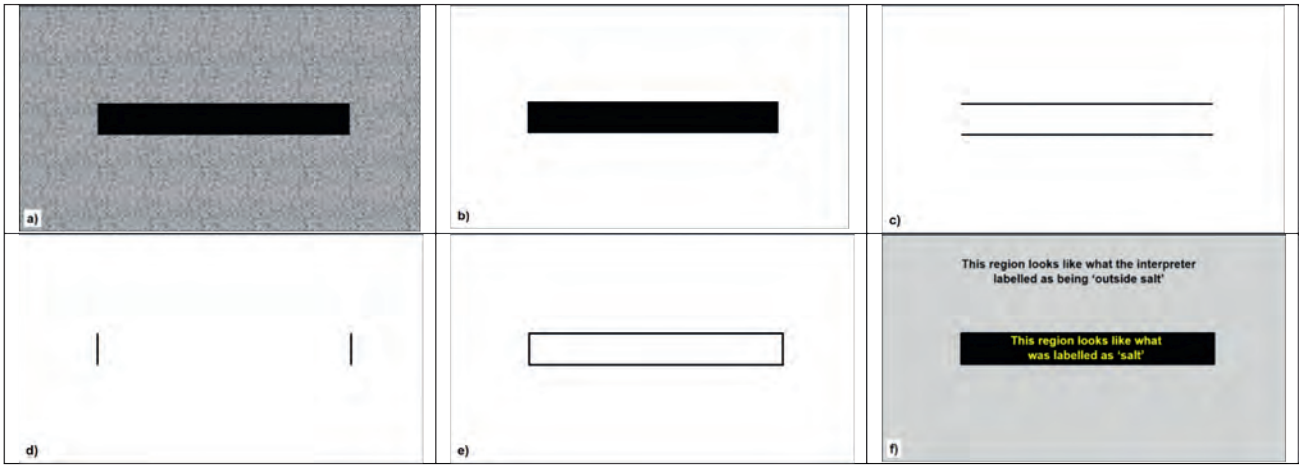


**Figure 3** Schematic workflow cartoon for possible steps in isolating and labelling a salt body: a) input image; b) threshold values to remove small numbers ('noise'); c) compute derivative of image b) in the vertical direction; d) compute derivative of image b) in the horizontal direction; e) add the derivatives from various directions to identify all edges; f) classify the regions 'inside' and 'outside' the object.

underlying physics of the moveout behaviour of the data (there is some 'theory' behind what we did). However, for ML, there is no underlying physical theory governing what the basis vectors turn out to be; the NN simply finds some descriptor that minimized the overall error in the network.

In recent years several other ML applications have appeared in the geophysical literature, for example: first-break picking (Sliz et al., 2021) and noise suppression (Martin et al., 2015; Klochikhina et al., 2020; Walpole et al., 2020). Figure 2 shows a common receiver gather from an ocean bottom node survey, where ML has been used to pick the first arrivals. A 50 m * 50 m shot 'carpet' was acquired over 3000 nodes with a 500 m * 500 m node spacing. Manual first-break picking was performed on 1% of the data so as to '*label*' the first breaks, and subsequent use of ML enabled picking of the first breaks on 650 million traces in a few hours. These picks were used as a guide for refraction waveform inversion.

Next, I'll consider a more challenging problem, namely identifying a salt boundary in a migrated image. Starting with several thousand small 2D image segments showing a top-salt (and/or base-salt) boundary and the previously manually interpreted (*labelled*) position of this boundary.We want to derive a processing flow to match the pixels in the 2D image at the boundary to the manually picked interpretation. This processing flow is essentially intended and designed to '*simplify and classify*' the input seismic data based on the supplied prior interpretations. This will be the *training* stage for building the neural network. The output will be a set of optimized processing parameters for shaping and matching data to their corresponding known manual interpretations based on various criteria which will be characterised by the ML optimization procedure.

Now, we move on to the inverse problem: take a new 2D image segment from a 3D image volume and ask if there is a salt boundary present in this segment of the data, and if so, where. The network assesses whether or not we have salt by applying the neural network processing flow with the previously derived optimized (trained) parameters, and if the cost function for the new 2D image segment is very low, we decide that salt is indeed present, and where in the 2D image it is located. Following this procedure, we would label every pixel in the new small image segment as being outside salt or within salt. Moving the small 2D analysis window around a new previously unseen 3D image could then classify the volume in terms of image pixels being outside of, or within, the salt body e.g. TGS's Kaggle competition (Kaggle, 2019; Kainkaryam et al., 2019; Milosavljevic, 2020).

In a more general sense, what we want to do is recognise an edge: this could be the top edge of a salt body, or in the context of handwriting recognition the edge of a number or letter, or the edge of a cat's face in a photo. An edge constitutes a discontinuity with respect to some background, so a first step could be to enhance and detect the location of changes. Figure 3a represents a 2D image segment, with background noise (representing the uninteresting bits) and a black rectangle representing the object we want to identify. First, we set a threshold on the data to zero out any small values (Figure 3b). We then compute the difference between two successive samples (a derivative) in the vertical direction so as to emphasize a sudden change in values – the vertical derivative emphasizes (detects) the *top* and *bottom* of the rectangle. Perhaps we then rectify the result to make all values positive, and also apply a non-linear scaling to the remaining numbers to force them to take on values between one and ten (Figure 3c). The derivative could be performed in other directions to help identify other edges: Figure 3d is the rectified *lateral* derivative which detects the *sides* of the rectangle. Performing the derivative for several directions and summing the results could help detect to edges with various dips (Figure 3e). In addition to finding the edges of the shape, we want to ascertain if numerical values inside or outside the shape have the characteristics of say 'salt' or 'not-salt'. In the geophysical context, we note that seismic data samples near the top of a salt body tend to be characterized by a lower-frequency waveform at the boundary and generally, for allochthonous salt bodies, also have little layered structure within the body. Conversely, the material outside such salt more often has a layered structure.

Yet another way of comparing subsets of pixels within the small 2D image segment would be to take a small 'pilot' image (representing what we are looking for: in this case an edge) and to convolve this small pilot image step-by-step with all pixels in the larger 2D image segment. This will enable us to obtain a correlation coefficient at every point in the image. If the small pilot image resembles a part of the 2D image segment, then the convolution will produce a large value of the correlation coefficient. Figure 4a shows the small pilot image (the black rectangle positioned near the top-left corner of the 2D image segment). The pilot image is moved pixel-by-pixel towards the right (Figure 4b), and the convolution repeated at each location. At the location in Figure 4c, the pilot coincides with some structure in the 2D image and a maximum in the convolution will occur. This convolutional procedure would then be repeated

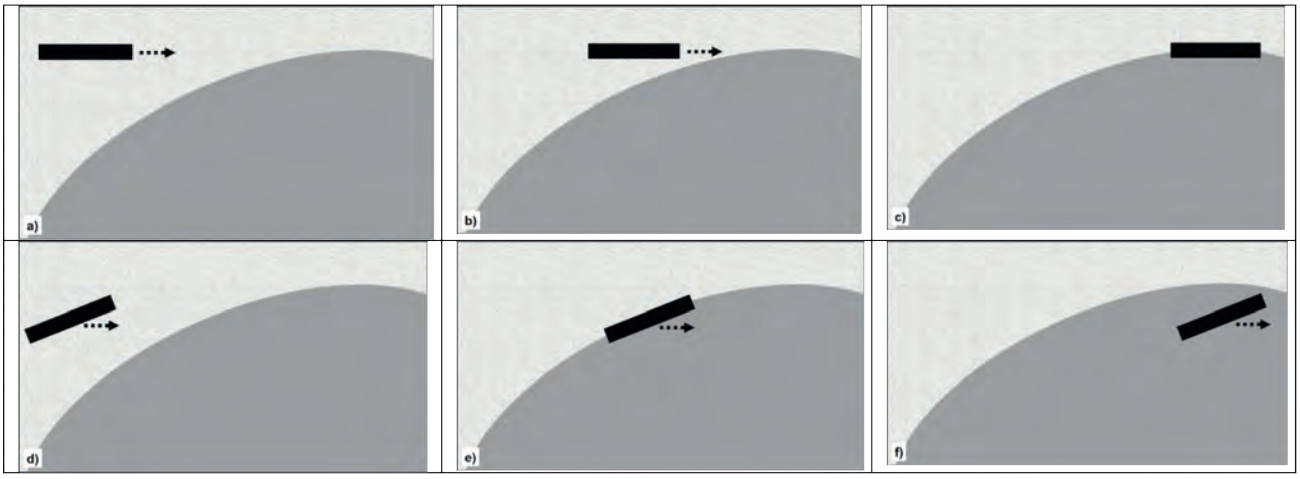


**Figure 4** Schematic workflow cartoon for detecting an interface via comparison with a pilot 'template' by correlating the pilot with all parts of the 2D image segment. In the top row, the pilot pixel template (black rectangle) slides to the right in small steps: at the far right (c) the pilot template sits on the interface and will give a high numerical value in the convolution. The lower row shows one of many dipping pixel sets which likewise are slid in small steps to the right, overlapping the interface at location (e). Once the pilot has been moved around all locations on the 2D image segment, the locations of the correlation maxima will outline the interface location.

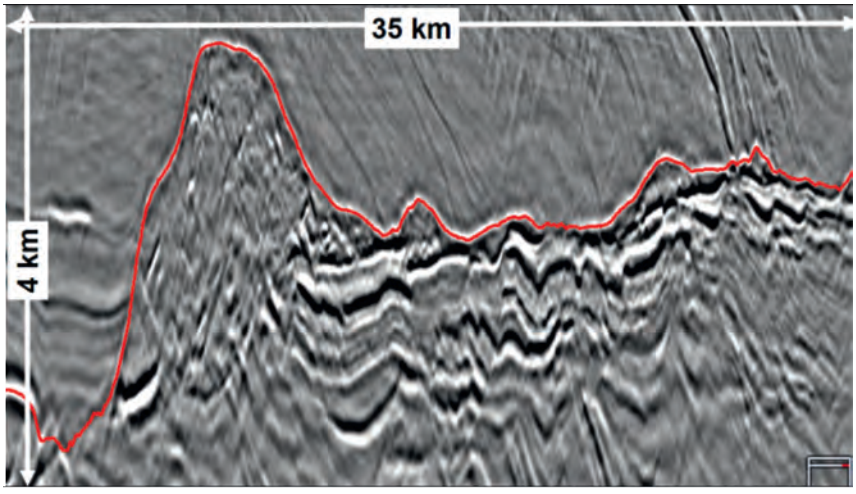


**Figure 5** Small preSDM image segment from a large 3D marine survey showing ML picking of the top salt surface indicated by the red line (courtesy of Xin Huang, ION).

after rotating the pilot through various angles, as indicated in Figure 4d-f (as we want to find interfaces with any orientation or dip). This approach would form part of a *'convolutional neural network'* (CNN) (e.g. Waldeland, 2018).

Figure 5 shows an example of using ML to pick the top of salt from a large multi-survey 3D marine project: this survey spans tens of thousands of square kilometres, hence automating picking of the salt is a great benefit for project turnaround.

Alternatively, we might take a completely different approach: within the 2D image segment, take a smaller group of pixels, say 3x3, and cross-correlate every 3x3 subset from the 2D image with all other possible 3x3 subsets. The covariance matrix of these correlations could then be used to identify which parts of the 2D image were similar to each other, thus helping to characterize regions as being inside or outside the salt body.

The above procedures can be improved by optimizing the parameters that were used: this is where the *training* step comes in again. In this case, the parameters to be optimized could be the width and height of the pilot rectangle, any tapers on its edges, whether or not we rectify the values, and whether or not we scale or equalize the values in the image being analysed, etc. Again, investigating all possible combinations of such parameters is a wholly impractical task for a person to undertake, but is trivial for an algorithm (given enough computing power).

We compare the location of the salt body estimated by the procedure with that provided by an interpreter. Setting up an iterative least-squares back-propagation workflow to adjust the parameters until the cost function is minimized will improve the estimation procedure. If this is done on many thousands of different examples where the human interpretation was provided, then the chances are that any new (and previously unseen example) will be correctly identified by then running the (now) optimized procedure. In this way, with an interpreter having labelled which parts of the training data sets were *inside* and which parts *outside* the salt body, the ML procedure should be able to label a salt body in the new, previously unseen data with a high degree of accuracy (Figure 3f). In the TGS Kaggle examples, the ML algorithms had success rates in the high 90 percentiles.

## Full waveform inversion in the context of ML

Yet another example of a misfit-minimization procedure that readers may be familiar with is that of full waveform inversion (FWI). In FWI we perform forward modelling of shot gathers for an initial velocity model, and iterate changing the model until these modelled shots closely resemble the actual field data (e.g. Lailly 1983; Tarantola 1984; Virieux et al., 2014; Brittan et al., 2013; Jones, 2018, 2019). The parameters involved in FWI are primarily the earth model parameters governing elastic wave propagation (velocity, anisotropy, density, attenuation, etc.), but may also include more mundane processing parameters, such as the mutes applied to the shot gathers, the tapers on the mutes, the amplitude scaling applied to the trace, band-pass filtering, offset-weighting, etc.

The difference between the modelled and field data constitute a 'residual'. A series of equations linking a change in parameters to a change in the residual is set up (referred to as the Jacobian), and these equations are used to update the parameters after each iteration of forward modelling until the '*cost function*' (sum of squares of the residual) is acceptably reduced. Conventionally, we do not invert for the general processing parameters (this is usually left to a user's experience and accepted best practice): hence in conventional FWI, the parameters are simply the values of velocity (etc.) in the cells of the earth model.

This description can be recast as an ML problem. In ML terminology, these iterations constitute the *training*: the training is performed using many thousands of field shot records, and the optimized parameters make up the best-estimate velocity model for the survey (and could in principle also include the other more general processing parameters).

The following idea is not what we ever actually do in practice, but is meant to be an indication of what could be achieved via ML. Assume that we had some new data that were not used in the training, but were missing their source and receiver coordinates, and in addition we were not sure if these data were even acquired over the same geological area. For an individual shot record selected from these new data, we could use our trained network to undertake forward modelling iterating over all possible source and receiver positions within the area where the training had occurred. If an acceptably small cost function was obtained, then this would imply that the new shot was recorded with the source and receiver locations corresponding to this minimum cost function. Repeating this procedure for all shot records from the new survey, if we obtained acceptably small cost functions, we could conclude that these new, previously unseen data were indeed acquired over the same geological area, and also what their likely source and receiver positions were.

Conversely, if the data came from a different region of the Earth, then processing them with the above-mentioned ML workflow would give rise to unacceptably large residuals and cost function and this would be a clear indication that the new data were not from the geological area used in the training.

If our budget was large enough, we could reproduce the ML-FWI training exercise using data from all over the world. Then, when presented with a new, hitherto unseen, dataset, we could run forward modelling with all the different sets of optimized parameters, and select the parameter set that gave the smallest cost function. This would give an indication of what possible earth model gave rise to these new data, as the new data was similar to one of the training sets. This latter observation could be used to estimate a more generic low wavenumber model pertaining to the new dataset which could then be used as a starting point for a full model estimation procedure for the new input data (e.g. Araya-Polo et al., 2019).

It can be inferred from the above descriptions that the training is perhaps the most important aspect of the ML process, and it would be totally impractical to apply ML using seismic data from every region on the planet. However, although it may be impractical to use ML alone instead of a more conventional FWI, tomography and migration workflow, ML is indeed being introduced into various aspects of these flows (e.g. Araya-Polo, et al., 2019; Øye and Dahl, 2019; Milosavljevic, 2020; Sun and Alkhalifah, 2020; Sliz et al., 2021).

## Discussion

The background to ML is fairly straightforward in terms of how the maths is set up to update parameters by back propagating prediction errors. However, the outcomes of an ML procedure in terms of what it can achieve are generally non-intuitive and perhaps beyond human (or at least a geophysicist's) comprehension. This is perhaps best demonstrated in facial recognition, where an appropriately trained ML procedure can recognize an individual's face even when presented with pictures of that person at different ages. The optimized parameters are likely to be somehow encoding for things such as distance between the eyes, distance of the eyes from the edge of the face, distance from the nose to the lips, etc. With enough hidden layers, and activation functions, it can be seen that information can be 'boiled down' to a limited number of key indicators that encode for enough information so as to identify (recognize) what we want. Humans have evolved to achieve this task effortlessly, but comprehending how an ML procedure accomplishes this is another matter.

For more general applications, if a NN has been adequately trained then deployment of a trained NN for a routine geophysical problem can be much more efficient and less time consuming than the standard deterministic or iterative techniques, especially in manpower-intensive tasks (e.g. first-break picking). This could be particularly useful in cases when near real time processing is required, such as velocity estimation while drilling for pore-pressure anomaly detection. Uncertainty estimation would be another fruitful avenue to explore: employing several different trained NNs could furnish a range of equi-probable solutions, which could then be used to estimate error bounds on a parameter.

Introductory tutorial material on geophysical applications of ML can be found in the EAGE online lecture by Waldeland (2018), and for an excellent series of lectures on the underlying mathematics of ML, I think that the four tutorials by Sanderson (2017) and the two lectures by Amini (2020) give a solid foundation.

## Acknowledgements

I would like to thank John Brittan, Carlos Calderon, Paul Brettwood, and Peter Rowbotham for helpful suggestions with the manuscript, and to ION for permission to publish this work.

## References

Amini, A. [2020] MIT Introduction to Deep Learning, MIT 6.S191. https://www.youtube.com/watch?v=njKP3FqW3Sk&list=PLtBw6njQRU-rwp5__7C0oIVt26ZgjG9NI&index=1 [accessed 15 June 2020].

Amini, A. [2020] Convolutional Neural Networks, MIT 6.S191. https://www.youtube.com/watch?v=iaSUYvmCekI&list=PLtBw6njQRU-rwp5__7C0oIVt26ZgjG9NI&index=3 [accessed 15 June 2020].

Araya-Polo, M., Farris, S. and Florez, M. [2019] Deep learning-driven velocity model building workflow. *Leading Edge*, **38**(11), 872A1-872A9.

Brittan, J., Bai, J., Delome, H., Wang, C. and Yingst, D. [2013] Full waveform inversion - The state of the art. *First Break*, **31**(10), 75-81.

Calderón-Macías, C., Sen, M.K. and Stoffa, P.L. [1997] Hopfield neural networks, and mean field annealing for seismic deconvolution and multiple attenuation. *Geophysics*, **62**(3), 992-1002.

Jones, I.F. [2018] *Velocities, Imaging, and Waveform Inversion (the Evolution of Characterising the Earth's Subsurface)*. EAGE, 234pp.

Jones, I.F. [2019] Tutorial: the mechanics of waveform inversion. *First Break*, **37**(5), 31-43.

Jones, I.F, Felicio, R., Vlassopoulou, A. and Hagen, C., [2019].Tutorial: tomographic Bayesian uncertainty estimation. *First Break*, **37**(9), 37-48.

Kaggle.com, [2019] TGS Salt Identification Challenge, Segment Salt Deposits Beneath the Earth's Surface. https://www.kaggle.com/c/tgs-salt-identification-challenge [accessed on 10 June 2020].

Kainkaryam, S., Ong, C., Sen, S., and Sharma, A. [2019] Crowdsourcing salt model building: Kaggle-TGS salt identification challenge. *81st EAGE Conference and Exhibition*, Extended Abstracts

Klochikhina, E., Crawley, S., Frolov, S., Chemingui, N. and Martin, T. [2020] Leveraging deep learning for seismic image denoising. *First Break*, **38**(7), 41-48.

Lailly, P. [1983] The seismic inverse problem as a sequence of before stack migrations: Conference on inverse scattering: Theory and application. *SIAM*, 206-220.

Martin, T., Saturni, C. and Ashby, P. [2015] Using machine learning to produce a global automated quantitative QC for noise attenuation. SEG Technical Program Expanded Abstracts, 34, 4790-4794.

Milosavljevic, A., [2020] Identification of Salt Deposits on Seismic Images Using Deep Learning Method for Semantic Segmentation. *ISPRS International Journal of Geo-Information*, **9**(1), https://www.mdpi.com/2220-9964/9/1/24 [accessed 10 June 2020].

Øye, O.K. and Dahl, E.K. [2019] Velocity Model Building from Raw Shot Gathers Using Machine Learning. *81st EAGE Conference and Exhibition*, Extended Abstracts.

Rumelhart, D.E., Hinton, G.E., and Williams, R.J. [1986] Learning representations by back-propagating errors. *Nature*, **323**(9), 533-536.

Russell, B. [2019] Machine learning and geophysical inversion - A numerical study. *Leading Edge*, **38**(7), 512-519.

Sanderson, G. [2017] But what is a Neural Network? | Deep learning, chapter 1. https://www.youtube.com/watch?v=aircAruvnKk&list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi&index=2&t=567s [accessed 15 June 2020].

Sanderson, G. [2017] Gradient descent, how neural networks learn | Deep learning, chapter 2. https://www.youtube.com/watch?v=IHZwWFHWa-w&list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi&index=2 [accessed 15 June 2020].

Sanderson, G. [2017] What is backpropagation really doing? | Deep learning, chapter 3. https://www.youtube.com/watch?v=Ilg3gGewQ5U&list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi&index=3 [accessed 15 June 2020].

Sanderson, G. [2017] Backpropagation calculus | Deep learning, chapter 4. https://www.youtube.com/watch?v=tIeHLnjs5U8&list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi&index=4 [accessed 15 June 2020].

Schleicher, K. [2018] The conjugate gradient method, *The Leading Edge*, **37**(4), 296-298,

Shewchuck, J. [1994] An introduction to the conjugate gradient method without the agonizing pain: School of Computer Science, Carnegie

Mellon University, http://www.cs.cmu.edu/~quake-papers/painless-conjugate-gradient.pdf [accessed 10 June 2020].

Sliz, K.S., Huang, R., Burley, T. and Berranger, I. [2021] A velocity model building strategy in northern Red Sea using FWI with OBN data. *SEG/DGS Workshop: Challenges & New Advances in Velocity Model Building*, 9-11 March 2021

Sun, B. and Alkhalifah, T. [2020] ML-misfit : Learning a robust misfit function for full-waveform inversion using machine learning ML-misfit. https://www.researchgate.net/publication/339127367_ML-misfit_Learning_a_robust_misfit_function_for_full-waveform_inversion_using_machine_learning [accessed 10 June 2020].

Tarantola, A. [1984] Inversion of Seismic Reflection Data in the Acoustic Approximation. *Geophysics*, **49**, 1259-1266.

Verschuur, D.J, [2020] Machine learning to replace or to augment the physics? *90th SEG Annual International Meeting*, Expanded Abstracts, W20.

Virieux, J., Asnaashari, A., Brossier, R., Métivier, L., Ribodetti, A. and Zhou, W. [2014] An introduction to full waveform inversion. In *Encyclopedia of Exploration Geophysics*. https://doi.org/10.1190/1.9781560803027.entry6

Waldeland, A.U. [2018] Seismic interpretation with deep learning. EAGE E-Lecture: https://www.youtube.com/watch?v=lm85Ap4OstM [accessed 15 June 2020].

Walpole, J., Hallett, T., Brown, E. and Brittan, J. [2020] Visual identification of noisy seismic records with machine learning. *82nd EAGE Conference and Exhibition*, Extended Abstracts.

Wikipedia. Glossary of artificial intelligence. https://en.wikipedia.org/wiki/Glossary_of_artificial_intelligence [accessed 15 June 2020].

Zhou, Z.Z., Cvetkovic, M., Xu, B. and Fontana, P. [2012] Analysis of a broadband processing technology applicable to conventional streamer data. *First Break*, 30(10), 77-82.