Removing the effects of free-surface from seismic data using convolutional neural networks – Part 1: Theory and sensitivity analysis

Mert S. R. Kiraz[†,‡], *Roel Snieder*[†] & *Jon Sheiman*

[†] *Center for Wave Phenomena, Colorado School of Mines, Golden CO 80401, USA*

[‡]*Corresponding author*

E-mail: mertkiraz@gmail.com; rsnieder@mines.edu; jonsheiman1@gmail.com

(January 19, 2023)

CWP-1008

Running head: **Removing the effects of free-surface using CNNs**

ABSTRACT

Multiple events reflect more than once in the subsurface, and they are considered as noise since they shadow the useful information about the subsurface. In marine seismic data, the strongest multiples are those which reflect at least once from the free-surface (or the air-water interface). The attenuation of this type of multiple has been extensively studied and numerous approaches have been used to suppress free-surface multiples. However, free-surface multiple attenuation is an expensive seismic data processing step that often results in removing primary events from the seismic data along with the free-surface multiples when primary and multiple events overlap. We present an algorithm to attenuate free-surface multiples using convolutional neural networks (CNNs) and show that data from the CNN-based free-surface multiple attenuation results give a correlation coefficient of 0.97 on average with the numerically modeled data without free-surface multiples. We train a network using subsets of the Marmousi and Pluto velocity models, and make predictions using subsets of the Sigsbee velocity model. We demonstrate the robustness of CNNs for free-surface multiple attenuation using three numerical examples and show that CNNs are able to attenuate surface-related multiples even in the presence of overlapping primary and multiple events without removing the primary reflections. The network processes a single trace at a time, and therefore is not sensitive to missing traces or to sparse data acquisition, as is the case for ocean-bottom nodes.

INTRODUCTION

Although the presence of multiples in seismic data once was a doubtful question and was not considered as a serious limitation (Dix, 1948; Ellsworth, 1948), there is no doubt now that the removal of multiples from seismic data is one of the most challenging problems in seismic data processing. In general, there are three main categories on which the multiple attenuation algorithms rely: (1) Periodicity - so that the multiples can be attenuated if they have a periodic behavior. To address this type of periodic multiple, one can use deconvolution, τ-p deconvolution, deconvolution after stack (Backus, 1959; Diebold and Stoffa, 1981; Yilmaz, 2001). (2) Moveout - so that the multiples can be attenuated if they have a moveout that is sufficiently different than that of the primaries. To address this type of multiple, one can use stacking, f-k filtering (Mayne, 1962; Yilmaz, 2001). (3) Data-driven techniques - so that the multiples can be attenuated based on the information recorded in seismic data. Although there are several ways to address this type of multiple (e.g., inverse scattering multiple attenuation (ISMA), interbed multiple prediction (IMP)), they are out of the scope of this paper (Weglein et al., 1997; Araújo et al., 2005), and in this category, we only focus on surface-related multiple elimination (SRME) method (Verschuur et al., 1992).

Multiple reflections are one of the most challenging problems in recorded marine seismic data since they contaminate the primary reflections. The multiples directly related to the free-surface (or the sea surface) have at least one downward reflection at the free-surface and are the most significant multiples to address. SRME (Verschuur et al., 1992) is a data-driven technique, and it predicts multiples by exploiting the physical relationship between multiple reflections and their sub-events. The arrival time of a particular surface multiple

can be predicted by convolving the traces that contain the multiples' sub-events. SRME has been widely used to predict and iteratively subtract the multiples from the seismic data, and it is one of the most effective tools in seismic processing. However, it is computationally demanding and it does not always guarantee perfect solutions. For instance, during the subtraction step, one can remove the primary reflections from the seismic data in addition to the unwanted multiple reflections because of the overlapping primary and multiple events. Additionally, a profound limitation of SRME is that it requires a dense source and receiver distribution which makes it computationally demanding and which makes missing data gaps a complication that must be dealt with. Although 2D versions of SRME can be computationally efficient, they fall short in the presence of complex subsurface requiring 3D spatial distribution of source and receiver distribution (Dragoset et al., 2010) (e.g., when multiples with strong cross-line raypath components present in the data). This condition becomes even more difficult to fulfill in 3D surveys.

There have been several attempts to address free-surface multiple attenuation. Riley and Claerbout (1976) present free-surface multiple attenuation theory applied to 1D data, and they also present an algorithm for surface-generated diffracted multiple elimination. Verschuur et al. (1992) show that free-surface multiples can be attenuated by predicting the multiples and subtracting them from the seismic data. Jakubowicz (1998) shows that interbed multiples can be removed using a technique that is an extension of SRME. There have been field data applications of SRME (Dragoset and MacKay, 1993; Verschuur et al., 1995; Kelamis and Verschuur, 2000; Dragoset and Željko Jeričević, 1998).

SRME removes the events that reflect at least once from the free-surface (or the air-water interface). To predict the surface-related multiples only, the measured field data are usually used and no information about the subsurface is needed. However, SRME

makes great requirements about the data acquisition. For a successful implementation of SRME, some of the requirements are; a dense distribution of the source and receiver wavefields, known source wavelet, interpolation/extrapolation of the data, and available near-offset data. Conventionally, an SRME processing flow follows; (i) Data pre-processing, (ii) Prediction of the surface-related multiples, (iii) Matching filtering (or estimation of wavelet), (iv) Adaptive subtraction.

We present and discuss a CNN-based approach to attenuate free-surface multiples. We train a network using 2D subsets of Marmousi velocity model (Versteeg, 1994), and Pluto velocity model (Stoughton et al., 2001), and make predictions using three subsets of Sigsbee velocity model (Paffenholz et al., 2002) with salt bodies in the subsurface. As the training input data, we use 2D data modeled *with* free-surface multiples using the finite-difference forward modeling, and also use 2D data modeled *without* free-surface multiples using the finite-difference forward modeling as the training output data. Although we use 2D subsets of the velocity models for the training, we use 1D convolutional filters to find the optimal weights for the network that relate the input and output datasets. After training the network using the input and output data, we make predictions using three new datasets which were not involved in the training process. We show that CNNs are able to make predictions that give a correlation coefficient (CC) of 0.97 on average between the numerically modeled results and the CNN predictions, and when the free-surface multiples overlap with the primary events, the CNN-based solution is able to remove the unwanted free-surface multiple reflections effectively (e.g., CC of 0.95) from the seismic data while preserving the primary reflections.

CNN ARCHITECTURE AND TRAINING

Machine learning (ML) is a subset of artificial intelligence and it uses mathematical and statistical methods to let machines and computers improve specified tasks with experience. Deep learning (DL) is a subset of ML and it is inspired by the structure of our brains and the interconnections between its neurons. DL uses multiple layers of computational units and every layer learns its own representation of the input data (Ekman, 2021).

[Figure 1 about here.]

Figure 1 shows a simple one-hidden-layer (and in total three-layer) neural network where the input layer, hidden layer, and the output layer have three neurons to illustrate the propagation of a neuron in the input layer through the network. In Figure 1, the black arrows indicate that the three neurons in the input layer are connected to the first neuron in the hidden layer. As the number of hidden layers increase, the network extracts more complex information, and the networks with many hidden layers are called "deep neural networks".

We obtain the output value of a neuron shown in Figure 1 by

$$a_j^k = \sigma\Big(\sum_{i=1}^{N} (a_i^{(k-1)} w_{ij}^k) + b_j^k \Big) , \tag{1}$$

where N denotes the number of neurons in the previous layer, a_j^k denotes the value a to be calculated of the jth neuron in the kth layer, $a_i^{(k-1)}$ denotes the value a of the ith neuron in the $(k-1)$th layer, w_{ij}^k denotes the weight w of the ith neuron which is connected to the jth neuron in the kth layer, b_j^k denotes the bias b of the jth neuron in the kth layer. *Training* is a process in ML such that we adjust the weights, w, and biases, b, by an iterative data-fitting process. During the start of the training process, a set of initial adjustable random weights

and biases are assigned. As the training progresses, the weights and biases are updated. Also, σ in Eq. (1) denotes the activation function. The activation function introduces the non-linearity in machine learning algorithms. If there is no non-linearity between layers, using several linear transformations of stack of layers, we only get linear transformations and are not able to solve complex problems (Géron, 2019; Ekman, 2021).

[Figure 2 about here.]

When each neuron of a layer is connected to every neuron in the other layer, the network is called a fully connected network. Figure 2 shows an example of a fully connected network with an input layer, hidden layer, and an output layer where the input layer, hidden layer and the output layer have 3 neurons. Convolutional neural networks (CNNs) are important building blocks in DL methods inspired by the brain's visual cortex and they have been widely incorporated in image recognition studies since the 1980's (Géron, 2019). CNNs are different from other DL algorithms because the neurons in the CNN architecture are connected to the region of pixels/neurons (receptive fields) in the input image as opposed to every individual pixels/neurons (for more detailed information about CNNs, we refer readers to the Appendix).

[Figure 3 about here.]

During the training phase, once an output of a network (or the output of each neuron in the output layer) is obtained, we compare the output of the network to the desired output (or the ground truth) by defining an error function. One of the most commonly used error functions is the mean-squared error (MSE) metric that measures the difference between the

predicted and the ground truth results,

$$E = \frac{1}{m}\sum_{i=1}^{m}(\hat{y}_i - y_i)^2 \,, \tag{2}$$

where $\hat{y}_i$ is the output of the network, y_i is the ground truth data for a given input data during the training, and m is the number of training examples. To update the parameters, the MSE function is minimized commonly using the gradient descent method by calculating the gradient of the error function with respect to the weights, and training CNNs is commonly done by backpropagation where the choice loss function and gradient descent optimization algorithm play essential roles (Bishop, 2006; Géron, 2019; Ekman, 2021).

Different DL/CNN methods and their subsets have been used in exploration seismic studies to address different problems. Siahkoohi et al. (2018) utilize CNNs to remove the free-surface multiples and numerical dispersion; Das et al. (2019) use CNNs to obtain an elastic subsurface model using recorded normal-incidence seismic data; Wu et al. (2019) use CNNs for 3D seismic fault segmentation; Almuteri and Sava (2021) use CNNs to address to ghost removal from seismic data; Kiraz and Snieder (2022) utilize CNNs for 1D wavefield focusing where the solution of the Marchenko equation is not needed to retrieve the Green's function once the network is trained. Recently, CNNs have been used to tackle free-surface multiples in various ways (Siahkoohi et al., 2018, 2019; Ovcharenko et al., 2021; Zhang et al., 2021; Liu-Rong et al., 2021). In this paper, our network predicts a single trace at a time, and this makes our algorithm insensitive to the data gaps in the acquisition and does not require a meticulous data-filling pre-processing step.

[Figure 4 about here.]

Network architecture

For the prediction of the free-surface multiple attenuated data, the CNN architecture we use consists of an input layer, 9 hidden 1D convolutional layers, and an output layer. The first and second, fourth and fifth, and seventh and eighth hidden layers have 32 filters; the third, sixth, and ninth hidden layers have 8 filters (see the Appendix for more information about the action of the filters). The length of the 1D convolutional window (kernel size) for the first 3 hidden layers is 1125 samples; for the following 5 hidden layers is 101 samples; and for the last hidden layer is 51 samples. Overall, the network consists of 1,773,697 total parameters all of which are trainable. Figure 3 shows a cartoon of the network architecture used for this study.

For the input layer and the hidden convolutional layers, we use the rectified linear unit (ReLU) activation function (Ekman, 2021) which computes the function $f(x) = max(0, x)$. The ReLU activation function helps optimize the model easily using the gradient-based methods (Géron, 2019). While minimizing the error function, a backpropagation (or a backward pass) (Ekman, 2021) is used to update the weights of the network. The derivative of the activation function is calculated at this step, and the derivative of the ReLU activation function does not suffer from "vanishing gradients" phenomena (Géron, 2019; Ekman, 2021). Figure 4(a) shows the ReLU activation function and Figure 4(b) shows its derivative. Lastly, the output layer does not have an activation function.

For all the layers, we use a stochastic gradient descent optimization algorithm with momentum along with the mean-squared loss function as the objective function (Géron, 2019). To prevent our model from overfitting, and for an accurate evaluation, we split the input data into training dataset, testing dataset, and validation dataset using 60%,

20%, and 20% ratios, respectively. The testing data are used to monitor the training of the network, and adjust hyperparameters if needed. We tune the hyperparameters using this testing dataset and compare the output to the ground truth. One should be careful at this step because we want the network to learn the relationship between the input and output training data pairs, and if we tune the hyperparameters only according to the testing dataset, the network would memorize the relationship between the input and output data pairs instead of learning. This would also introduce the risk of overfitting, and the network would perform poorly when we use a different dataset which is not involved in the training process. After defining the network parameters (or hyperparameters), we use 100 epochs (iterations) to train the network. Figure 5 shows the learning curve of our network where the blue curve denotes the validation data and the red curve denotes training data.

[Figure 5 about here.]

The choice of hyperparameters in this paper is based on trial and error. After changing a set of hyperparameters, we evaluate the performance of the network by comparing the predictions to the expected results, by observing the training and validation curves in Figure 5) to make sure that we are not overfitting the data (one of the most common ways to check overfitting is to observe the training and validation curves during training (Géron, 2019; Ekman, 2021)).

Network training

We create 192 input shot gathers where each gather consists of 600 traces (total of 115,200 input traces) using 2D finite-difference forward modeling through the models with free-surface multiples, and 192 output shot gathers where each gather consists of 600 traces

(total of 115,200 output traces) using 2D finite-difference forward modeling through the models without free-surface multiples. Throughout the forward modeling for each model, 600 receivers are placed along 2.4 km horizontal extent with the receiver sampling of 4 m, and are located at 8 m depth. We use one source per model, and the coordinates of the source are given as $x_s = 25$ m and $z_s = 7$ m.

For the training model selection, we use the Marmousi and Pluto velocity models. Figure 6(a) shows the Marmousi velocity model and Figure 6(b) shows the Pluto velocity model. We extract 36 sub-models from the Marmousi model and 156 sub-models from the Pluto model. We also add 640 m of water layer to the top of each model with 1500 m/s velocity. Each one of 192 models used for training has 2.4 km horizontal extent, and 2.4 km depth which includes the water column, 600 receivers, and a single shot location. Figure 7 shows 36 sub-models of the Marmousi velocity model used for the training, and Figure 8 shows 156 sub-models of the Pluto velocity model used for the training. For each velocity model, we use a density model that is a scaled version of the velocity model by a factor two (and the units are converted from km/s to g/cm^3), and has a water layer of 640 m with the velocity of 1500 m/s. Figure 9 shows 1 of 192 velocity and density models used during the data generation for training. Figure 9(a) shows a training velocity sub-model extracted from Marmousi velocity model. Figure 9(b) shows a training density sub-model scaled from the velocity model shown in Figure 9(a). Scaling the velocity model by two and adding the water layer into the density model creates a range of reflection coefficients that changes from approximately 0.33 to 0.89. The minimum reflection coefficient at the water bottom is set to mimic the seafloor conditions which is comparable to the seafloor reflection coefficient given by Kim (2004). Figure 10 shows 5 of 192 shot gathers of the input and output data displayed next to each other used for training after removing the direct arrival. We remove

the direct arrival from the recorded data during the training and prediction steps. Figure 10(a) shows 5 of 192 shot gathers of the data modeled with free-surface multiples which are used as the training input, and Figure 10(b) shows 5 of 192 shot gathers of the data modeled without free-surface multiples which are used as the training output. Figure 10(a) shows the first appearance of free-surface multiples at each shot gather starting around 1.7 s in the first trace of a shot gather, and ending around 2.3 s in time. Although not as strong, the shot gathers also have other types of multiples (e.g., top and/or bottom salt peg-legs, internal).

[Figure 6 about here.]

[Figure 7 about here.]

[Figure 8 about here.]

[Figure 9 about here.]

[Figure 10 about here.]

[Figure 11 about here.]

NUMERICAL EXAMPLES AND CNN PREDICTION

For the numerical simulation, we use three different sub-models of the Sigsbee velocity model which are not used in the training process and are unknown to the network to demonstrate our CNN-based solution for free-surface multiple attenuation.

For the first example, Figure 11 shows a subset of the Sigsbee velocity model which consists of relatively flat subsurface structures along with a part of a salt body located at the deeper parts of the model. Similar to the training models, 600 receivers are placed along 2.4 km horizontal extent with the receiver sampling of 4 m at a depth of 8 m. The coordinates of the source are given as $x_s = 25$ m and $z_s = 7$ m. The water column is 640 m deep with a velocity of 1500 m/s and a density of 1 gr/cm^3. The sea bottom for this example also has a reflection coefficient of approximately 0.33.

Figure 12(a) shows the data modeled with free-surface multiples using the velocity model shown in Figure 11, and Figure 12(b) shows the waveforms modeled without free-surface multiples using the velocity model shown in Figure 11. Figure 12(a) shows the input data, and starting around 1.7 s in time at 0 km in offset, and ending around 2.3 s in time at 2.4 km in offset, the first multiple caused by the free-surface is located. Additionally, starting around 2.55 s in time at 0 km in offset, and ending around 2.75 s in time at 1.5 in km offset, another multiple caused by the free-surface is present. Although not as strong as these two events, there are other multiples arriving after the first multiple which are caused by the free-surface. Figure 12(b) shows the desired output, the ground truth. The free-surface multiples described in Figure 12(a) are not present in this figure because of the excluded free-surface effects. For a successful CNN network training, we expect the CNN predictions to be similar to the ground truth (Figure 12(b)). After training the network, Figure 12(c) shows our CNN-based free-surface multiple attenuation results. Figure 12(d) shows the difference between the ground truth data (Figure 12(b)) and our CNN-based free-surface multiple attenuation results (Figure 12(c)).

We measure the accuracy of the CNN-based free-surface multiple attenuation results by calculating trace-by-trace CC's between the ground truth and the CNN prediction. Figure

13 shows the CC's between the ground truth (Figure 12(b)) and the CNN prediction (Figure 12(c)), and the average CC is 0.98.

Figure 14 shows several trace comparisons of the input data (solid blue line), output data (or ground truth) (solid red line), the CNN predictions (solid black line), and the difference between the ground truth and the CNN prediction (solid green line) shown in Figure 12 for the zero-, mid-, and far-offsets. Figure 14(a) shows the trace used as an input to the network which is the zero-offset trace extracted from Figure 12(a) (solid blue line), the ground truth that is the expected result from the CNN prediction which is the zero-offset trace extracted from Figure 12(b) (solid red line), the predicted zero-offset trace using our CNN-based free-surface multiple attenuation operator extracted from Figure 12(c) (solid black line), and the difference zero-offset trace between the ground truth and our CNN-based free-surface multiple attenuation operator extracted from Figure 12(d) (solid green line). In Figure 14(a), the two strongest multiples caused by the free-surface in the input trace (solid blue line) at times around 1.7 s and 2.7 s have been successfully attenuated in the CNN prediction trace (solid black line) which matches the ground truth trace (solid red line). Figure 14(b) shows the mid-offset (offset 1.2 km) trace used as an input to the network from Figure 12(a) (solid blue line), the ground truth that is the expected result from the CNN prediction which is the mid-offset (offset 1.2 km) trace extracted from Figure 12(b) (solid red line), the predicted mid-offset (offset 1.2 km) trace using our CNN-based free-surface multiple attenuation operator extracted from Figure 12(c) (solid black line), and the difference mid-offset trace between the ground truth and our CNN-based free-surface multiple attenuation operator extracted from Figure 12(d) (solid green line). In Figure 14(b), the strongest multiple caused by the free-surface in the input trace (solid blue line) at time around 1.85 s has been successfully attenuated in the CNN prediction trace (solid

black line) which matches the ground truth trace (solid red line). Lastly, Figure 14(c) shows the far-offset (offset 2.3 km) trace used as an input to the network from Figure 12(a) (solid blue line), the ground truth that is the expected result from the CNN prediction which is the far-offset (offset 2.3 km) trace extracted from Figure 12(b) (solid red line), the predicted far-offset (offset 2.3 km) trace using our CNN-based free-surface multiple attenuation operator extracted from Figure 12(c) (solid black line), and the difference far-offset trace between the ground truth and our CNN-based free-surface multiple attenuation operator extracted from Figure 12(d) (solid green line). In Figure 14(c), the strongest multiple caused by the free-surface in the input trace (solid blue line) at time around 2.35 s has been successfully attenuated in the CNN prediction trace (solid black line) which matches the ground truth trace (solid red line). The difference traces for the near-, mid-, and far-offset traces indicate a small amplitude mismatch between the ground truth and the CNN-based free-surface multiples. Figure 14 shows that the CNN-based free-surface multiple attenuation is able to attenuate the free-surface multiples despite their different arrival times due to the offset change. The CNN-based free-surface multiple attenuation is able to preserve the primary events after removing the unwanted multiples from the data. Additionally, the phase change in the input trace is successfully accounted for through the CNN prediction. The presence of the free-surface changes the phase and the amplitude spectrum by the interference with the ghost wave. Thus, the algorithm suppresses multiples and does the deghosting simultaneously(Riley and Claerbout, 1976; Amundsen and Zhou, 2013).

[Figure 12 about here.]

[Figure 13 about here.]

For the second example, Figure 15 shows another subset of the Sigsbee velocity model which consists of relatively flat subsurface structures with a fault near the far-offsets, and a salt body located at the deeper parts of the model. Similar to the training models, 600 receivers are placed along 2.4 km horizontal extent with the receiver sampling of 4 m, and are located at 8 m depth. The coordinates of the source are given as $x_s = 25$ m and $z_s = 7$ m. The water column has 640 m depth with a velocity of 1500 m/s. The sea bottom for this example also has a reflection coefficient of approximately 0.33.

[Figure 14 about here.]

Figure 16(a) shows the data modeled with free-surface multiples using the velocity model shown in Figure 15, and Figure 16(b) shows the data modeled without free-surface multiples using the velocity model shown in Figure 15. Figure 16(a) shows the input data, and starting around 1.7 s in time at 0 km in offset, and ending around 2.3 s in time at 2.4 km in offset, the first multiple caused by the free-surface appears. The challenge in this example is that the first event caused by the free-surface overlaps with a primary event around 1.5 km in offset and 1.85 s in time in Figure 16(a). Conventional SRME techniques require an adaptive subtraction step in which the presence of overlapping primary and multiple events might result in removal or reduction of the primary energy. Although not as strong as the first event caused by the free-surface, there are other multiples arriving after the first multiple which are caused by the free-surface. Figure 16(b) shows the desired output. The events described in Figure 16(a) caused by the free-surface are not present in this figure because of the excluded free-surface. For a successful CNN network training, we expect the CNN predictions to be similar to the ground truth (Figure 16(b)). After training the network, Figure 16(c) shows our CNN-based free-surface multiple attenuation shot gather.

Figure 16(d) shows the difference between the ground truth data (Figure 16(b)) and our CNN-based free-surface multiple attenuation shot gather (Figure 16(c)).

[Figure 15 about here.]

Figure 17 shows the CC's between the ground truth (Figure 16(b)) and the CNN prediction (Figure 16(c)), and the average CC is 0.95 which indicates a strong similarity between the ground truth data and the CNN prediction. Figure 18 shows several trace comparisons of the input data (solid blue line), output data (or ground truth) (solid red line), the CNN predictions (solid black line), and the difference between the ground truth and the CNN prediction (solid green line) shown in Figure 16 for the zero-, mid-, and far-offsets. Figure 18(a) shows the trace used as an input to the network which is the zero-offset trace extracted from Figure 16(a) (solid blue line), the ground truth that is the expected result from the CNN prediction which is the zero-offset trace extracted from Figure 16(b) (solid red line), the predicted zero-offset trace using our CNN-based free-surface multiple attenuation operator extracted from Figure 16(c) (solid black line), and the difference zero-offset trace between the ground truth and our CNN-based free-surface multiple attenuation operator extracted from Figure 16(d) (solid green line). In Figure 18(a), the strongest multiple caused by the free-surface in the input trace (solid blue line) at time around 1.7 s has been successfully attenuated in the CNN prediction trace (solid black line) which matches the ground truth trace (solid red line). Figure 18(b) shows the mid-offset (offset 1.5 km) trace used as an input to the network from Figure 16(a) (solid blue line), the ground truth that is the expected result from the CNN prediction which is the mid-offset (offset 1.5 km) trace extracted from Figure 16(b) (solid red line), the predicted mid-offset (offset 1.5 km) trace using our CNN-based free-surface multiple attenuation operator extracted from Figure 16(c) (solid

black line), and the difference mid-offset trace between the ground truth and our CNN-based free-surface multiple attenuation operator extracted from Figure 16(d) (solid green line). However, in Figure 18(b), the strongest multiple caused by the free-surface in the input trace (solid blue line) at time around 1.9 s overlaps with a primary event. Although the free-surface reflection is embedded in the high-amplitude energy around 1.9 s in Figure 18(b), the CNN prediction is able to remove the free-surface multiple without removing the primary events. The CNN prediction trace (solid black line) closely matches the ground truth trace (solid red line) in Figure 18(b). Lastly, Figure 18(c) shows the far-offset (offset 2.3 km) trace used as an input to the network from Figure 16(a) (solid blue line), the ground truth that is the expected result from the CNN prediction which is the far-offset (offset 2.3 km) trace extracted from Figure 16(b) (solid red line), the predicted far-offset (offset 2.3 km) trace using our CNN-based free-surface multiple attenuation operator extracted from Figure 16(c) (solid black line), and the difference far-offset trace between the ground truth and our CNN-based free-surface multiple attenuation operator extracted from Figure 16(d) (solid green line). In Figure 18(c), the strongest multiple caused by the free-surface in the input trace (solid blue line) at time around 2.35 s has been successfully attenuated in the CNN prediction trace (solid black line) which matches the ground truth trace (solid red line). The difference traces for the near-, mid-, and far-offset traces indicate a small amplitude mismatch between the ground truth and the CNN-based free-surface multiples. Figure 18 shows that the CNN-based free-surface multiple attenuation is able to predict and remove the free-surface multiples and preserve the primary reflection even when the primary reflection and the free-surface multiple arrive simultaneously.

[Figure 16 about here.]

[Figure 17 about here.]

[Figure 18 about here.]

So far in this paper, we used different sub-models from Sigsbee velocity model for prediction with a reflection coefficient of around 0.33 at the sea-bottom. However, to test the accuracy of the CNN-based free-surface multiple attenuation prediction, without changing the training data, we use a sub-model from Sigsbee velocity model for prediction with a reflection coefficient of approximately 0.5 at the sea-bottom (we create the reflection coefficient of approximately 0.5 at the sea-bottom by scaling the density model accordingly and adding the water layer). Figure 19(a) shows the data modeled with free-surface multiples using the velocity model shown in Figure 11, but this time with a reflection coefficient of approximately 0.5 at the sea-bottom, and Figure 19(b) shows the waveforms modeled without free-surface multiples using the velocity model shown in Figure 11 with a reflection coefficient of approximately 0.5 at the sea-bottom. Figure 19(a) shows the input data, and starting around 1.7 s in time at 0 km in offset, and ending around 2.3 s in time at 2.4 km in offset, the first multiple caused by the free-surface (or the first-order free-surface multiple) is located. Additionally, starting around 2.55 s in time at 0 km in offset, and ending around 2.75 s in time at 1.5 in km offset, another multiple caused by the free-surface is present. Note that in Figure 19(a) the multiples are of higher amplitude than in Figure 12(a) because of the higher reflection coefficient at the sea-bottom. Figure 19(b) shows the desired output. Figure 19(c) shows our CNN-based free-surface multiple attenuation results. Figure 19(d) shows the difference between the ground truth data (Figure 19(b)) and our CNN-based free-surface multiple attenuation results (Figure 19(c)). This example shows that our CNN-based method attenuates the free-surface multiples with different

reflection coefficients at the sea-bottom.

[Figure 19 about here.]

[Figure 20 about here.]

Figure 20 shows the CC's between the ground truth (Figure 19(b)) and the CNN prediction (Figure 19(c)), and the average CC is 0.98 which indicates a strong similarity between the ground truth data and the CNN prediction. Figure 21 shows the trace view comparison of the input, ground truth, and predicted data with a reflection coefficient of 0.5 at the sea bottom. Figure 21(a) shows a zero-offset trace comparison with the first-order free-surface multiple arriving around 1.7 s, and some other free-surface multiples arriving at 2.55 s and 2.7 s. Figure 21(b) shows a mid-offset (1.2 km) trace comparison with the first-order free-surface multiple arriving around 1.85 s, and some other free-surface multiples arriving at 2.4 s and 2.65 s. Figure 21(c) shows a far-offset (2.4 km) trace comparison with the first-order free-surface multiple arriving around 2.35 s. The difference traces for the near-, mid-, and far-offset traces indicate a small amplitude mismatch between the ground truth and the CNN-based free-surface multiples. Figure 21 shows that our CNN-based free-surface multiple attenuation method successfully attenuates the surface-related multiples caused by a different reflection coefficient at the sea-bottom.

[Figure 21 about here.]

DISCUSSION AND CONCLUSIONS

The examples presented in this paper show that the CNN-based free-surface multiple attenuation is able to remove the multiples caused by the free-surface so that the CC between

the numerically modeled data without the free-surface multiples and the CNN predictions is found to be around 0.97 on average. The change in the arrival time of the free-surface multiples with offset is correctly predicted by the CNN and the multiples are removed from the shot gathers without removing the primary events even when the primary and multiple reflections overlap. Although the data obtained by the CNN prediction mostly match the numerically modeled data, for late times of the shot gathers (i.e., for times around 2.75 s), there are mismatches between the CNN prediction and the ground truth. This is due to the length of the 1D convolutional window (kernel size) running into the zero-padding towards the end of each trace to match the input and output data size in the network (see Figure 22). Kernel size (which changes between 51, 101, and 1125 samples in our case) determines the sliding window length to perform convolution along the sample, and to match the size of the input and output samples, we use zero-padding at the beginning and the end of each trace of a layer (see Figure 22). This causes both the early and late arrivals of the CNN predictions to have more mismatches than the rest of the data. Although the same happens for the early times of the shot gathers (i.e., for times around 0.75 s), there are no events recorded at such early times in the shot gathers. Although the CNN prediction and the desired output have a strong similarity (see Figures 13, 17, and 20), Figures 12(d), 16(d), and 19(d) show that there is an overall amplitude mismatch between the desired output and CNN prediction. The network is able to make predictions where the free-surface multiples are mostly attenuated, however for a better amplitude match between the desired output and CNN prediction, further hyperparameter tuning may be required (e.g., different combinations of the activation functions, number of hidden layers, number of filters, kernel sizes, strides, iterations (or epochs), choice of the loss function).

As we describe above, in the conventional SRME workflow, it is hard to accurately know

if the predicted multiples have the precise amplitude information with the matching filtering, and/or if the adaptive subtraction step only removes the multiples without removing the primary events from the data. Additionally, the dense source/receiver distribution requirement makes SRME computationally expensive and hard to implement in case of a sparse data acquisition (e.g., ocean-bottom node (OBN) data acquisition). Since our CNN-based approach does not require dense source/receiver distribution, once the network is trained it can be applied to any type of acquisition where we have either 1 trace or 1,000,000 traces available. It also does not require an additional step to subtract the multiples from the data because the network predicts the data without free-surface multiples.

The CNN-based free-surface multiple attenuation is able to remove the surface-related multiples caused by different sea-bottom reflection coefficients. To further test the accuracy of our CNN-based free-surface multiple attenuation method, we can also test the conditions where we have variations in the source signature, source and receiver depth, water column depth, and application to the OBN and/or carbon capture, utilization and storage (CCUS) data. However, the variations of these parameters fall out of the scope of this paper and will be discussed in a future paper.

The use of 1D convolutional filters makes our CNN-based free-surface multiple attenuation computationally efficient (e.g., training takes 81 minutes and prediction of a trace takes 2.4 ms). Additionally, trace-by-trace attenuation of free-surface multiples does not require a dense spatial distribution of sources and receivers. Our CNN-based method is suitable for irregular and sparse source/receiver acquisition geometry, and the accuracy of the prediction does not rely upon the availability of the near-offset and neighboring traces (which is one of the important requirements for the conventional SRME algorithms).

Our numerical examples show that free-surface multiples for subsurface models with variable velocity and density profiles with salt structures can be attenuated using CNNs once a network is trained. We show that when the primary and multiple reflections overlap, our CNN-based free-surface multiple attenuation method successfully removes the multiples while preserving the primary reflections. Unlike the conventional SRME methods, our algorithm does not require modeling and subtraction of the multiples from the full dataset for free-surface multiple attenuation, and eliminates the risk of removing the primary energy from the recorded seismic data by providing a fast and effective alternative to the existing free-surface multiple attenuation methods. Our algorithm does not require dense spatial distribution of source and receivers, and it is also independent of the availability of the near-offset and neighboring traces.

ACKNOWLEDGMENTS

We thank Khalid Almuteri from the Center for Wave Phenomena for his help with the CNN setup. This work is supported by the Consortium Project on Seismic Inverse Methods for Complex Structures at the Colorado School of Mines. The finite-difference forward modeling examples in this paper were generated using the Madagascar software package (Fomel et al., 2013). Machine learning implementations were completed using the TensorFlow source library (`https://www.tensorflow.org/`).

APPENDIX: CONVOLUTIONAL NEURAL NETWORKS

Figure 22 shows a sketch of a two-layer 1D convolutional neural network with a three-sample 1D convolutional window (kernel). In Figure 22, there are 4 neurons, $a_1^{(1)}$, ..., $a_4^{(1)}$ in *layer* 1,

and 4 neurons, $a_1^{(2)}$, ..., $a_4^{(2)}$ in *layer* 2 where the subscript denotes the neuron number, and the superscript denotes the layer number. Note that the term *neuron* is usually used in the context of a sample. For example, if we consider the neurons in *layer* 1 or *layer* 2 as a column vector, each neuron corresponds to an element in that column vector. There is also a three-sample temporal window in Figure 22 denoted with w_1, w_2, and w_3. The temporal window (or kernel) size is defined by the user in CNNs, and this kernel is initialized with random weights, which will later be updated during the training process. We slide this kernel from the beginning to the end of the sample in *layer* 1. As the kernel slides down, we calculate a centered dot product between *layer* 1 and assign the result of the dot product to the neurons in *layer* 2. We preserve the size of the neurons in each layer by zero padding at the beginning and the end of the neurons, and the zero padding is represented by the red neurons with zeros in Figure 22. Therefore, the result of the zero-padded centered convolution in *layer* 2 matches the shape of the input trace in *layer* 1. Note that if we do not use zero padding, the number of neurons in *layer* 2 does not match the number of neurons in *layer* 1.

As the kernel slides down, we calculate a centered dot product between zero-padded *layer* 1 and the kernel, and assign the result of the dot product to the neurons in *layer* 2 as;

$$\begin{aligned} a_1^{(2)} &= \sigma\Big((a_1^{(1)}w_2) + (a_2^{(1)}w_3) + b_1\Big)\,, \\ a_2^{(2)} &= \sigma\Big((a_1^{(1)}w_1) + (a_2^{(1)}w_2) + (a_3^{(1)}w_3) + b_1\Big)\,, \\ a_3^{(2)} &= \sigma\Big((a_2^{(1)}w_1) + (a_3^{(1)}w_2) + (a_4^{(1)}w_3) + b_1\Big)\,, \\ a_4^{(2)} &= \sigma\Big((a_3^{(1)}w_1) + (a_4^{(1)}w_2) + b_1\Big)\,, \end{aligned} \tag{3}$$

where b_1 denotes the bias term and the subscript denotes the number of output traces for a layer, and σ denotes the activation function. The trainable parameters of the network

shown in Figure 22 are w_1, w_2, w_3, and b_1. However, due to the zero padding around the beginning and the end of the sample, the prediction around these edges tends to be less accurate than the ones around the middle of the sample.

[Figure 22 about here.]

The number of output traces for a layer is another hyperparameter that needs to be determined. Figure 23 shows a cartoon of a 1D convolutional neural network for a two-layer network where the number of output traces for a layer is three and kernel size is also three. Similar to the example with one filter (see Figure 22), as the three filters slide down, we calculate a centered dot product between *layer* 1 and each filter, and assign the result of the dot product to the neurons in *layer* 2. Figure 23(a) shows the first filter acting on *layer* 1, Figure 23(b) shows the second filter acting on *layer* 1, and Figure 23(c) shows the third filter acting on *layer* 1. Figure 23(d) shows the input layer, *layer* 1, and the outputs of three filters on *layer* 2. The trainable parameters of the network shown in Figure 23 are w_1, w_2, w_3, w_4, w_5, w_6, w_7, w_8, w_9, b_1, b_2, and b_3.

In summary, for each input trace to a layer and each output trace from a layer, an independent filter is applied, the number of such filters is the number of input traces times the number of output traces. For each output trace, these results are summed over the input traces, a bias term is added (one bias for each output trace), and then the activation function is applied as;

$$a_j^k = \sum_i^N \sigma(w_{ji} * a_i^{k-1} + b_j) \ , \tag{4}$$

where a_i^{k-1} is the ith output trace from layer $k-1$, a_j^k is the jth output trace from layer

k, w_{ji} is the filter connecting input trace i to output trace j, and b_j is the bias for output trace j, $*$ denotes convolution in time, and N is the number of neurons in the previous layer.

Networks with multiple filters extract more information from the training input and output data pairs. However, increasing the number of filters also increases the computational demand of the network. Note that in Figures 22 and 23 the kernels slides down one sample at a time and this parameter (which is one of the hyperparameters) is called *stride*. For a 1D output in the next layer (*layer* 3), we define three different sets of weights which will be applied to the output on *layer* 2 shown in Figure 23(d), and the linear combination of these three different filtered outputs will form the output in *layer* 3 after adding the bias term and applying the activation function.

Although there is more hyperparameter selection and tuning involved during the training process, the illustration and description of each one of them are out of the scope of this paper. For more detailed information about DL methods and CNNs, we refer the reader to Ekman (2021); Géron (2019); Bishop (2006); Higham and Higham (2019).

[Figure 23 about here.]

REFERENCES

Almuteri, K., and P. Sava, 2021, A convolutional neural network approach for ghost removal: First International Meeting for Applied Geoscience & Energy Expanded Abstracts, 2550–2554.

Amundsen, L., and H. Zhou, 2013, Low-frequency seismic deghosting: GEOPHYSICS, **78**, WA15–WA20.

Araújo, F. V., A. B. Weglein, P. M. Carvalho, and R. H. Stolt, 2005, Inverse scattering series for multiple attenuation: An example with surface and internal multiples: SEG Technical Program Expanded Abstracts 1994, 1039–1041.

Backus, M. M., 1959, Water reverberations—their nature and elimination: GEOPHYSICS, **24**, 233–261.

Bishop, C. M., 2006, Pattern Recognition and Machine Learning: Springer.

Das, V., A. Pollack, U. Wollner, and T. Mukerji, 2019, Convolutional neural network for seismic impedance inversion: GEOPHYSICS, **84**, R869–R880.

Diebold, J. B., and P. L. Stoffa, 1981, The traveltime equation, tau-p mapping, and inversion of common midpoint data: GEOPHYSICS, **46**, 238–254.

Dix, C. H., 1948, The existence of multiple reflections: GEOPHYSICS, **13**, 49–50.

Dragoset, B., and S. MacKay, 1993, Surface multiple attenuation and sub-salt imaging: Exploration Geophysics, **24**, 463–472.

Dragoset, B., E. Verschuur, I. Moore, and R. Bisley, 2010, A perspective on 3D surface-related multiple elimination: GEOPHYSICS, **75**, 75A245–75A261.

Dragoset, W. H., and Željko Jeričević, 1998, Some remarks on surface multiple attenuation: GEOPHYSICS, **63**, 772–789.

Ekman, M., 2021, Learning deep learning: Theory and practice of neural networks, com-

puter vision, NLP, and transformers using TensorFlow: Addison-Wesley Professional.

Ellsworth, T. P., 1948, Multiple reflections: GEOPHYSICS, **13**, 1–18.

Fomel, S., P. Sava, I. Vlad, Y. Liu, and V. Bashkardin, 2013, Madagascar: open-source software project for multidimensional data analysis and reproducible computational experiments: Journal of Open Research Software, **1**, e8.

Géron, A., 2019, Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow, 2nd edition: O'Reilly Media, Inc.

Higham, C. F., and D. J. Higham, 2019, Deep Learning: An Introduction for Applied Mathematicians: SIAM Review, **61**, 860–891.

Jakubowicz, H., 1998, Wave equation prediction and removal of interbed multiples: SEG Technical Program Expanded Abstracts, 1527–1530.

Kelamis, P. G., and D. J. Verschuur, 2000, Surface-related multiple elimination on land seismic data—strategies via case studies: GEOPHYSICS, **65**, 719–734.

Kim, G.Y., R. M. B. D., 2004, Sediment types determination using acoustic techniques in the Northeastern Gulf of Mexico: Geosciences Journal, **8**, 95–103.

Kiraz, M. S. R., and R. Snieder, 2022, Marchenko focusing using convolutional neural networks: Second International Meeting for Applied Geoscience & Energy, 1930–1934.

Liu-Rong, T., J. Jin-Sheng, R. Hao-Ran, Y. Yue-Ming, and W. Bang-Yu, 2021, The separation of seismic surface-related multiples based on CAE-SAGAN: First International Meeting for Applied Geoscience & Energy Expanded Abstracts, 2914–2918.

Mayne, W. H., 1962, Common reflection point horizontal data stacking techniques: GEOPHYSICS, **27**, 927–938.

Ovcharenko, O., A. Baumstein, and E. Neumann, 2021, Surface-related multiple elimination through orthogonal encoding in the latent space of convolutional autoencoder: First

International Meeting for Applied Geoscience & Energy Expanded Abstracts, 1355–1359.

Paffenholz, J., J. Stefani, B. McLain, and K. Bishop, 2002, Sigsbee_2a synthetic subsalt dataset - image quality as function of migration algorithm and velocity model error: European Association of Geoscientists & Engineers, 2214–4609.

Riley, D., and J. Claerbout, 1976, 2-D multiple reflections: GEOPHYSICS, **41**, 1942.

Siahkoohi, A., M. Louboutin, R. Kumar, and F. J. Herrmann, 2018, Deep-convolutional neural networks in prestack seismic: Two exploratory examples: SEG Technical Program Expanded Abstracts, 2196–2200.

Siahkoohi, A., D. J. Verschuur, and F. J. Herrmann, 2019, Surface-related multiple elimination with deep learning: SEG Technical Program Expanded Abstracts, 4629–4634.

Stoughton, D., J. Stefani, and S. Michell, 2001, 2D elastic model for wavefield investigations of subsalt objectives, deep water Gulf of Mexico: SEG Technical Program Expanded Abstracts, 1269–1272.

Verschuur, D. J., A. J. Berkhout, and P. G. Kelmis, 1995, Estimation of multiple scattering by iterative inversion, part II: Examples on marine and land data: SEG Technical Program Expanded Abstracts, 1470–1473.

Verschuur, D. J., A. J. Berkhout, and C. P. A. Wapenaar, 1992, Adaptive surface-related multiple elimination: GEOPHYSICS, **57**, 1166–1177.

Versteeg, R., 1994, The Marmousi experience: Velocity model determination on a synthetic complex data set: The Leading Edge, **13**, 927–936.

Weglein, A. B., F. A. Gasparotto, P. M. Carvalho, and R. H. Stolt, 1997, An inverse-scattering series method for attenuating multiples in seismic reflection data: GEOPHYSICS, **62**, 1975–1989.

Wu, X., L. Liang, Y. Shi, and S. Fomel, 2019, Faultseg3d: Using synthetic data sets to train

an end-to-end convolutional neural network for 3D seismic fault segmentation: GEOPHYSICS, **84**, IM35–IM45.

Yilmaz, O., 2001, Seismis Data Analysis: Society of Exploration Geophysicists.

Zhang, D., M. de Leeuw, and E. Verschuur, 2021, Deep learning-based seismic surface-related multiple adaptive subtraction with synthetic primary labels: First International Meeting for Applied Geoscience & Energy Expanded Abstracts, 2844–2848.

LIST OF FIGURES

1 A sketch of a simple one-hidden-layer neural network with an input layer, hidden layer, and an output layer. For illustration purposes, the input layer, hidden layer, and the output layer have 3 neurons. 33

2 A sketch of a fully connected neural network with an input layer, hidden layer, and an output layer. The input layer, hidden layer and the output layer have 3 neurons. 34

3 Cartoon of the 1D network used for surface-related multiple attenuation on 2D data. The height of each trace corresponds to the sample size in our experiment which is 1126 samples. The numbers located below each trace represents the number of convolutional filters used (e.g., 1, 32, 32, 8, ...). Each blue area visually represents the kernel size used for each layer in the network (e.g., 1125, 101, 51). 35

4 (a) ReLU activation function. (b) Derivative of the ReLU activation function. 36

5 Learning curve for the convolutional neural network trained using the training and validation datasets. 37

6 (a) Marmousi velocity model. (b) Pluto velocity model. 38

7 Subsets of Marmousi velocity model used for training. 39

8 Subsets of Pluto velocity model used for training. 40

9 (a) One of the training velocity sub-models extracted from Marmousi velocity model. (b) One of the training density sub-models scaled from the velocity model in (a). 41

10 (a) 5 of 192 shot gathers of the input training data (*with* a free surface). (b) 5 of 192 shot gathers of the output training data (*without* a free surface). . 42

11 A sub-model of Sigsbee velocity model used for the prediction. 43

12 (a) Numerically modeled shot gather with the free-surface multiples (input data) obtained using the velocity model given in Figure 11. (b) Numerically modeled shot gather without the free-surface multiples (output data/ground truth) obtained using the velocity model given in Figure 11. (c) Predicted shot gather without the free-surface multiples using CNNs. (d) Difference between the ground truth data (Figure 12(b)) and the CNN prediction (Figure 12(c)). 44

13 Trace-by-trace calculated correlation coefficient between the ground truth data (Figure 12(b)) and the CNN prediction (Figure 12(c)). 45

14 Trace comparison of the input (solid blue line), ground truth (solid red line), CNN prediction (solid black line), and the difference between the ground truth and the CNN prediction (solid green line) data from the zero-, mid-, and far-offsets, respectively, extracted from the data shown in Figure 12. (a) Comparison of the zero-offset traces. (b) Comparison of the mid-offset (offset 1.2 km) traces. (c) Comparison of the far-offset (offset 2.3 km) traces. . . . 46

15 A sub-model of Sigsbee velocity model used for the prediction. 47

16 (a) Numerically modeled shot gather with the free-surface multiples (input data) obtained using the velocity model given in Figure 15. (b) Numerically modeled shot gather without the free-surface multiples (output data/ground truth) obtained using the velocity model given in Figure 15. (c) Predicted shot gather without the free-surface multiples using CNNs. (d) Difference between the ground truth data (Figure 16(b)) and the CNN prediction (Figure 16(c)) 48

17 Trace-by-trace calculated correlation coefficient between the ground truth data (Figure 16(b)) and the CNN prediction (Figure 16(c)). 49

18 Trace comparison of the input (solid blue line), ground truth (solid red line), CNN prediction (solid black line), and the difference between the ground truth and the CNN prediction (solid green line) data from the zero-, mid-, and far-offsets, respectively, extracted from the data shown in Figure 16. (a) Comparison of the zero-offset traces. (b) Comparison of the mid-offset (offset 1.5 km) traces. (c) Comparison of the far-offset (offset 2.3 km) traces. . . . 50

19 (a) Numerically modeled shot gather with the free-surface multiples (input data) with a reflection coefficient of 0.5 at the sea-bottom obtained using the velocity model given in Figure 11. (b) Numerically modeled shot gather without the free-surface multiples (output data/ground truth) obtained using the velocity model given in Figure 11. (c) Predicted shot gather without the free-surface multiples using CNNs. (d) Difference between the ground truth data (Figure 19(b)) and the CNN prediction (Figure 19(c)). 51

20 Trace-by-trace calculated correlation coefficient between the ground truth data (Figure 19(b)) and the CNN prediction (Figure 19(c)). 52

21 Trace comparison of the input (solid blue line), ground truth (solid red line), CNN prediction (solid black line), and the difference between the ground truth and the CNN prediction (solid green line) data from the zero-, mid-, and far-offsets, respectively, extracted from the data shown in Figure 19. (a) Comparison of the zero-offset traces. (b) Comparison of the mid-offset (offset 1.2 km) traces. (c) Comparison of the far-offset (offset 2.3 km) traces. . . . 53

22 A sketch of a zero-padded 1D convolutional neural network for a two-layer network where the number of output traces for a layer is three. 54

23 A sketch of a 1D convolutional neural network for a two-layer network along with three three-sample temporal windows. (a) The first filter and its output on *layer 2*. (b) The second filter and its output on *layer 2*. (c) The third filter and its output on *layer 2*. (d) Output of three three-sample windows on *layer 2*. 55

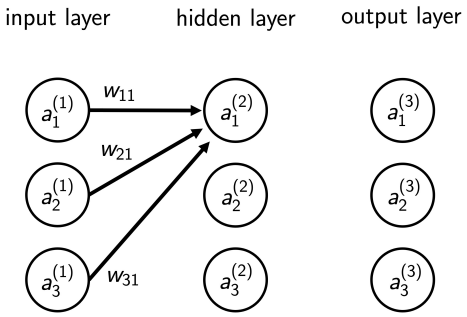


Figure 1: A sketch of a simple one-hidden-layer neural network with an input layer, hidden layer, and an output layer. For illustration purposes, the input layer, hidden layer, and the output layer have 3 neurons.

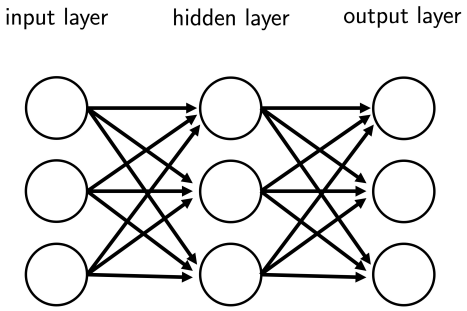


Figure 2: A sketch of a fully connected neural network with an input layer, hidden layer, and an output layer. The input layer, hidden layer and the output layer have 3 neurons.

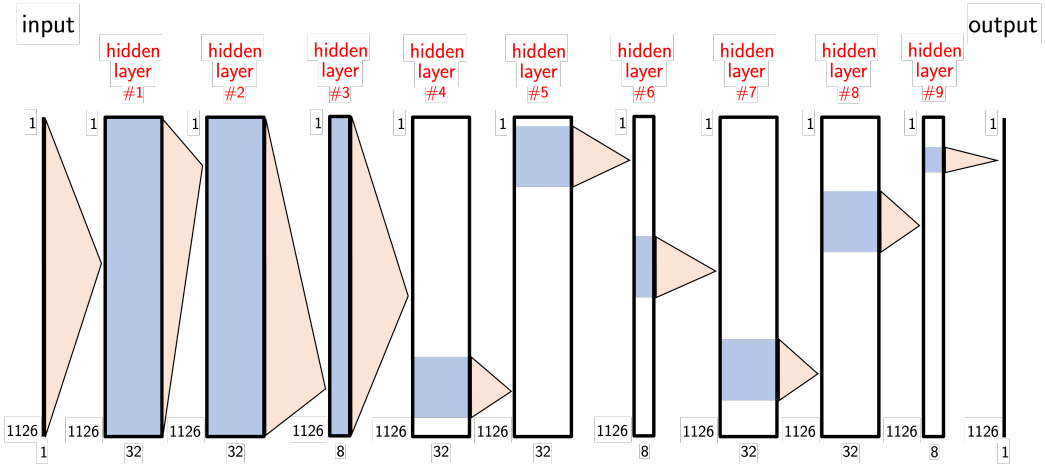
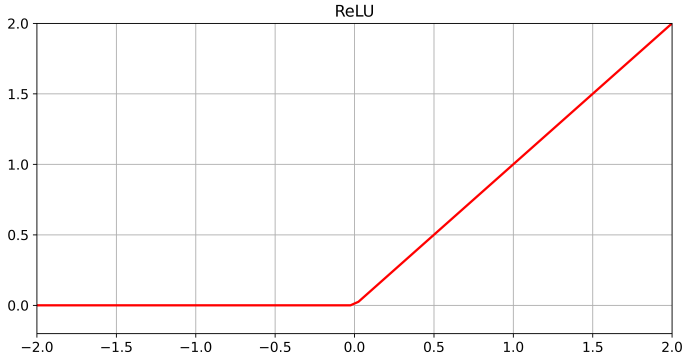
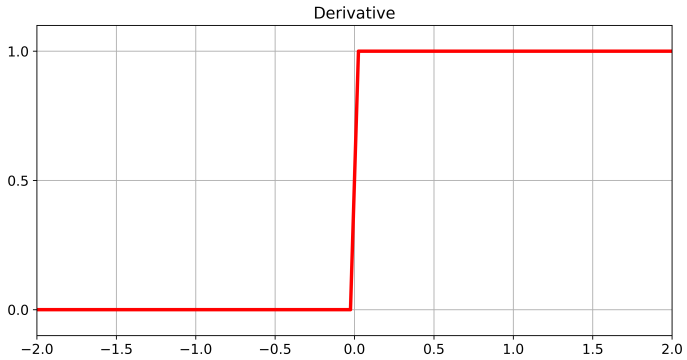


Figure 3: Cartoon of the 1D network used for surface-related multiple attenuation on 2D data. The height of each trace corresponds to the sample size in our experiment which is 1126 samples. The numbers located below each trace represents the number of convolutional filters used (e.g., 1, 32, 32, 8, ...). Each blue area visually represents the kernel size used for each layer in the network (e.g., 1125, 101, 51).



(a)



(b)

Figure 4: (a) ReLU activation function. (b) Derivative of the ReLU activation function.

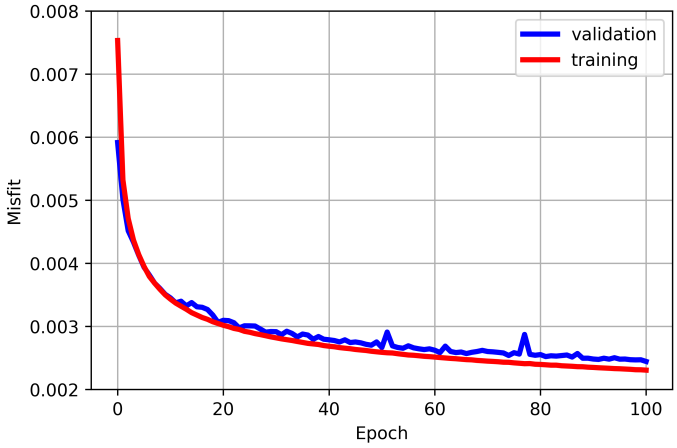
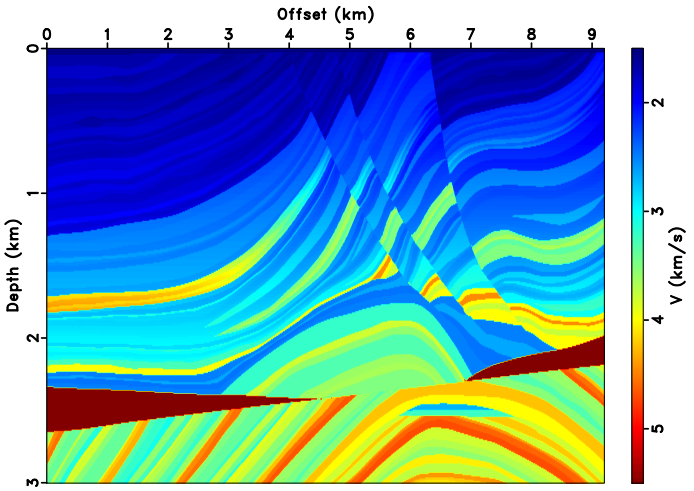
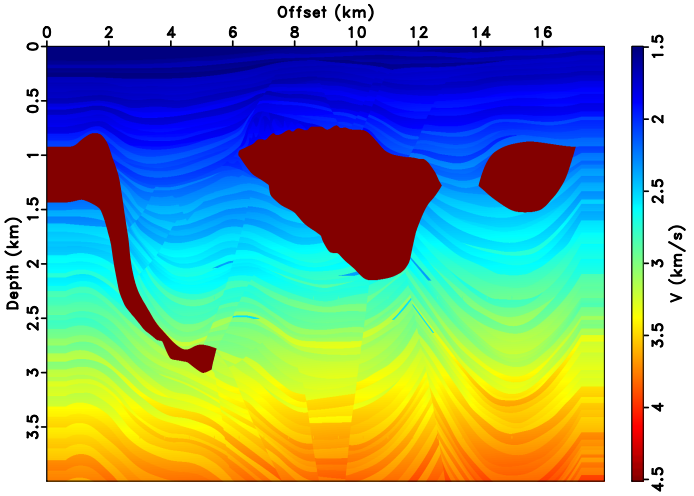


Figure 5: Learning curve for the convolutional neural network trained using the training and validation datasets.



(a)



(b)

Figure 6: (a) Marmousi velocity model. (b) Pluto velocity model.

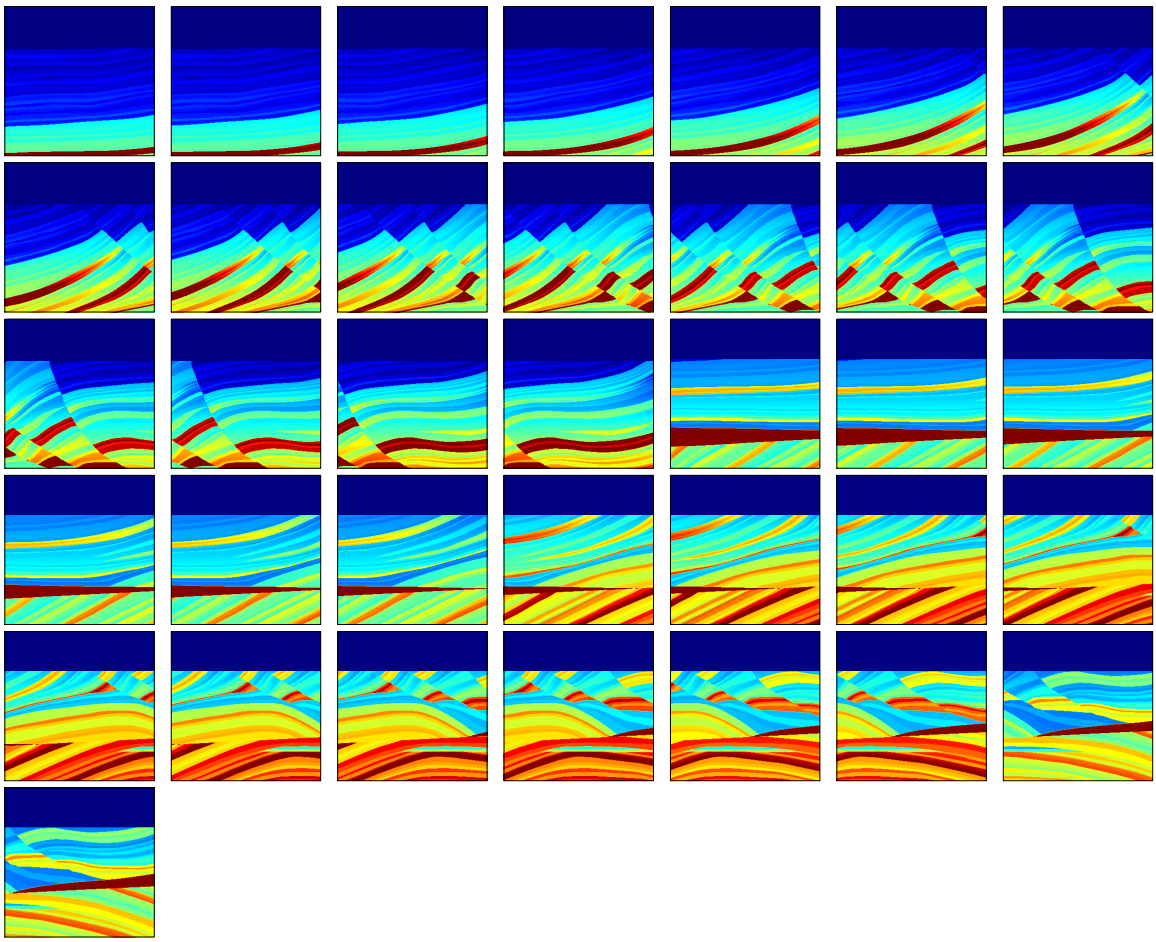


Figure 7: Subsets of Marmousi velocity model used for training.

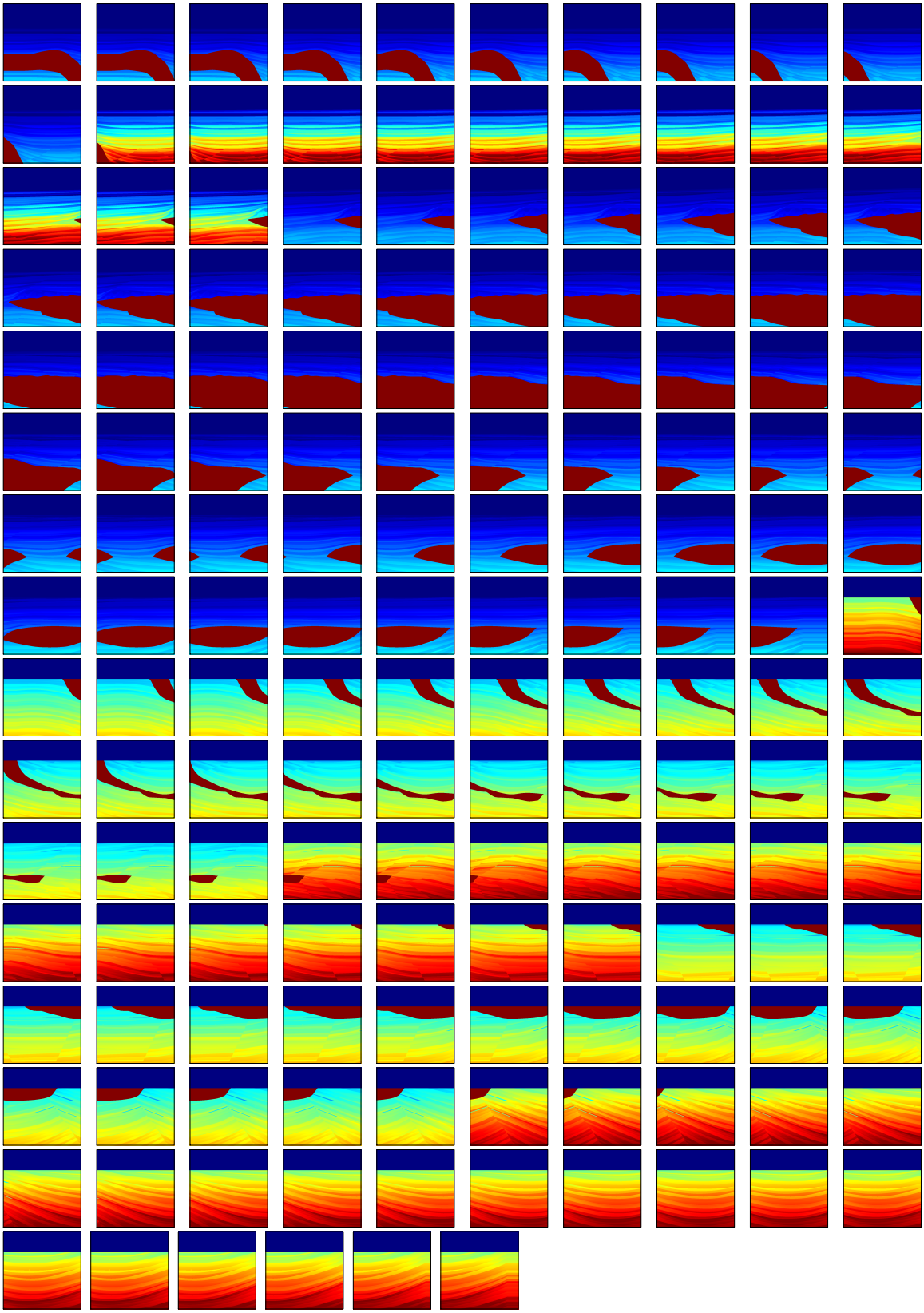
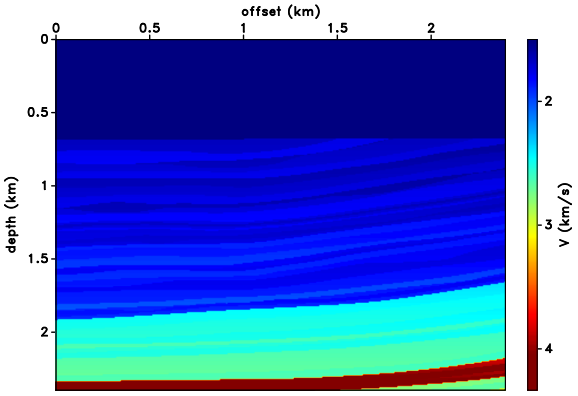
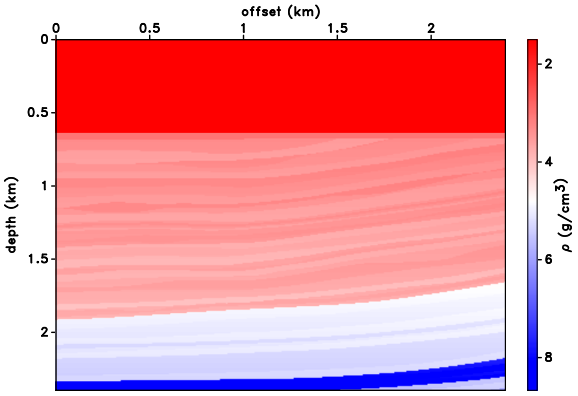


Figure 8: Subsets of Pluto velocity model used for training.

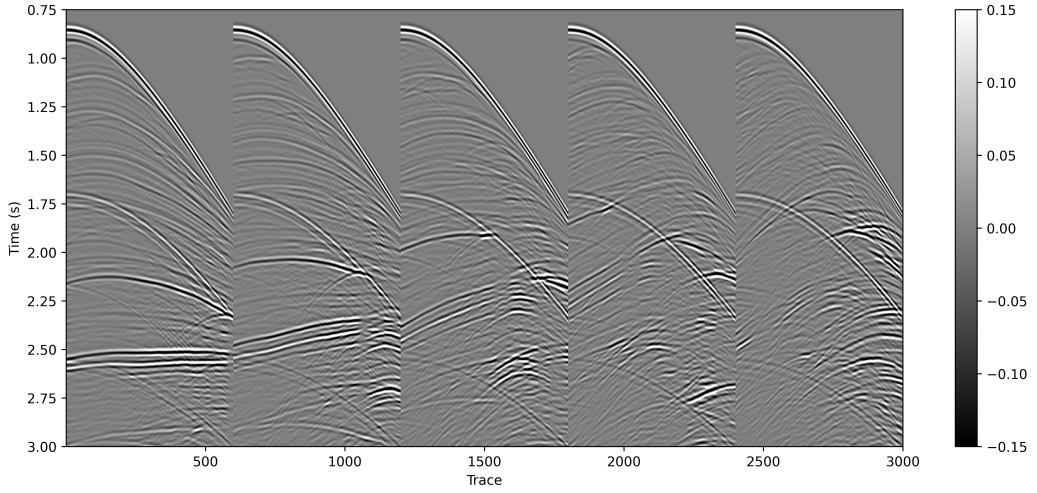


(a)

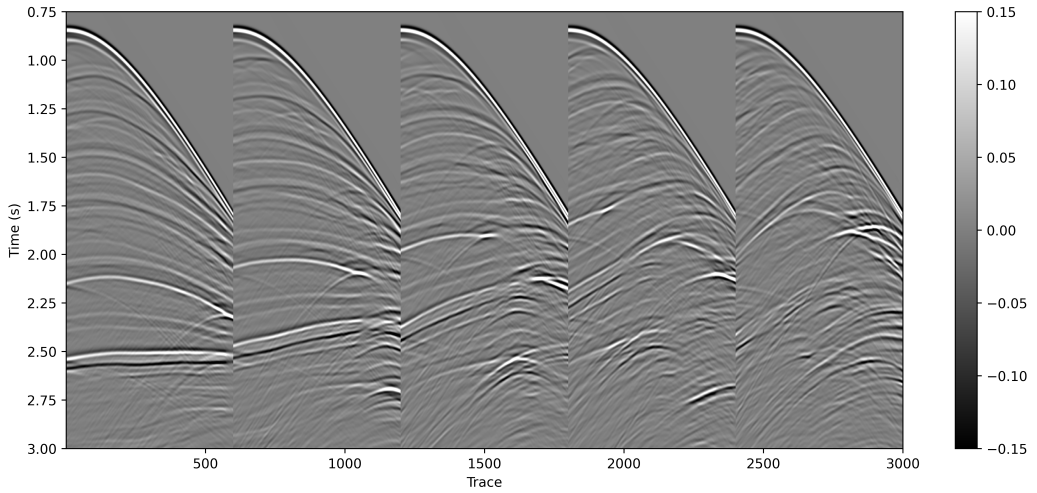


(b)

Figure 9: (a) One of the training velocity sub-models extracted from Marmousi velocity model. (b) One of the training density sub-models scaled from the velocity model in (a).



(a)



(b)

Figure 10: (a) 5 of 192 shot gathers of the input training data (*with* a free surface). (b) 5 of 192 shot gathers of the output training data (*without* a free surface).

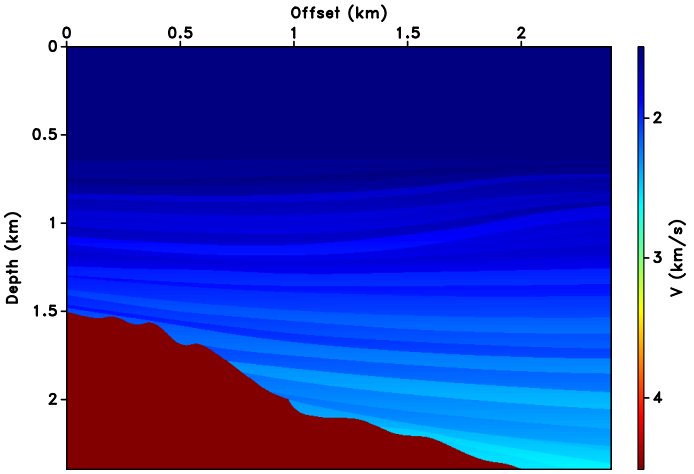


Figure 11: A sub-model of Sigsbee velocity model used for the prediction.

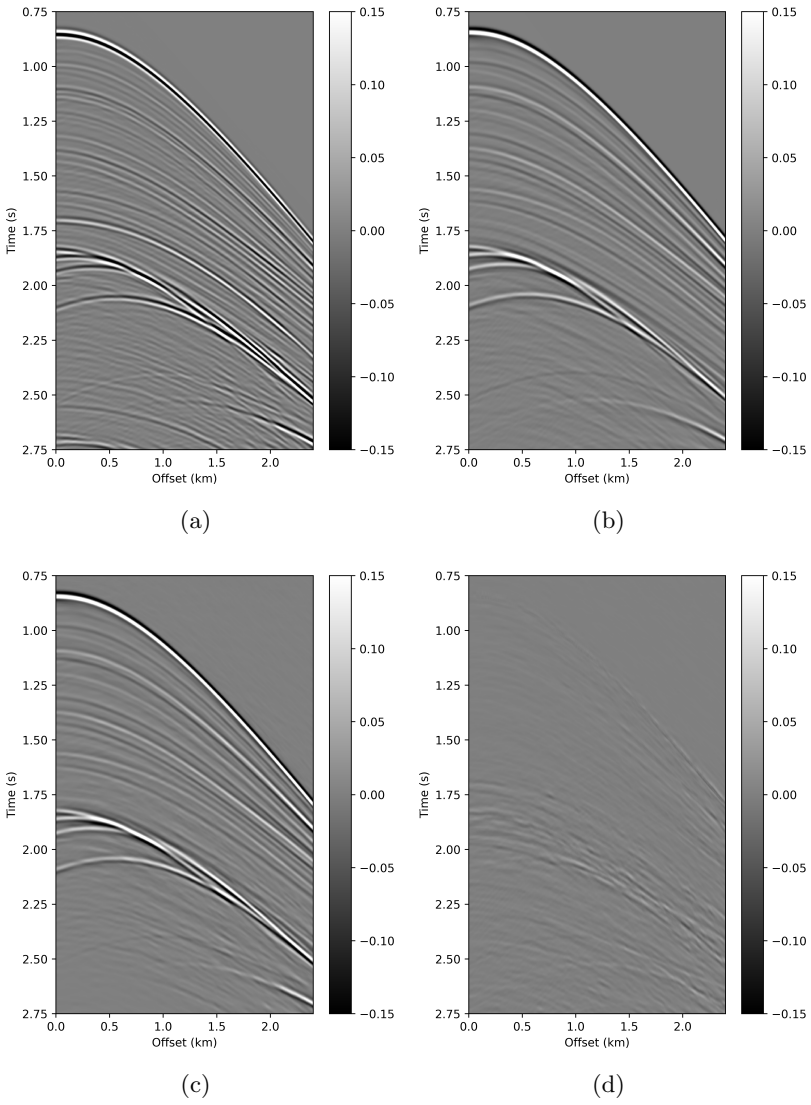


Figure 12: (a) Numerically modeled shot gather with the free-surface multiples (input data) obtained using the velocity model given in Figure 11. (b) Numerically modeled shot gather without the free-surface multiples (output data/ground truth) obtained using the velocity model given in Figure 11. (c) Predicted shot gather without the free-surface multiples using CNNs. (d) Difference between the ground truth data (Figure 12(b)) and the CNN prediction (Figure 12(c)).

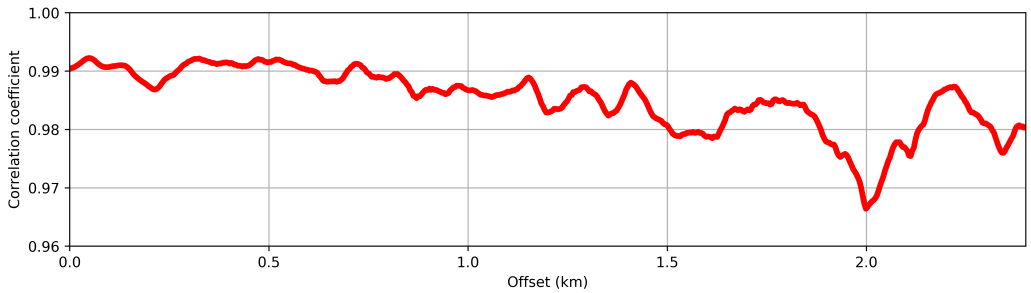
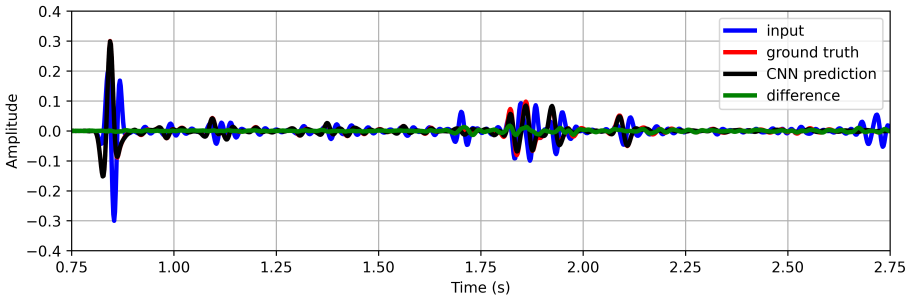
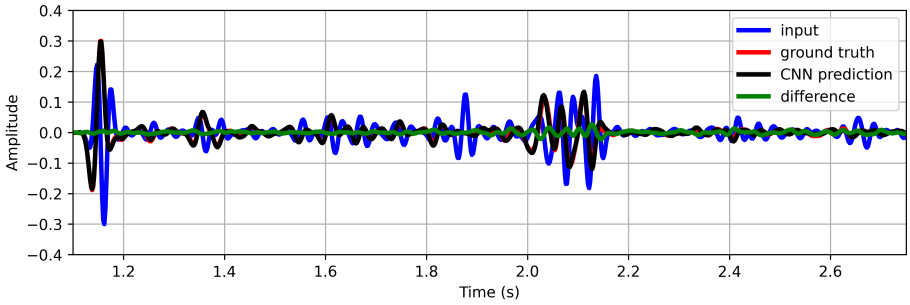


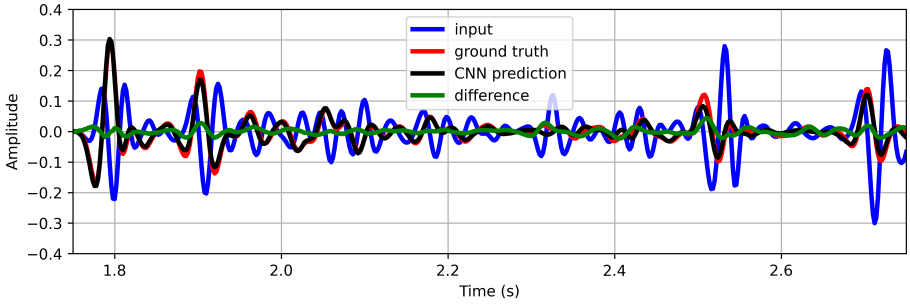
Figure 13: Trace-by-trace calculated correlation coefficient between the ground truth data (Figure 12(b)) and the CNN prediction (Figure 12(c)).



(a)



(b)



(c)

Figure 14: Trace comparison of the input (solid blue line), ground truth (solid red line), CNN prediction (solid black line), and the difference between the ground truth and the CNN prediction (solid green line) data from the zero-, mid-, and far-offsets, respectively, extracted from the data shown in Figure 12. (a) Comparison of the zero-offset traces. (b) Comparison of the mid-offset (offset 1.2 km) traces. (c) Comparison of the far-offset (offset 2.3 km) traces.

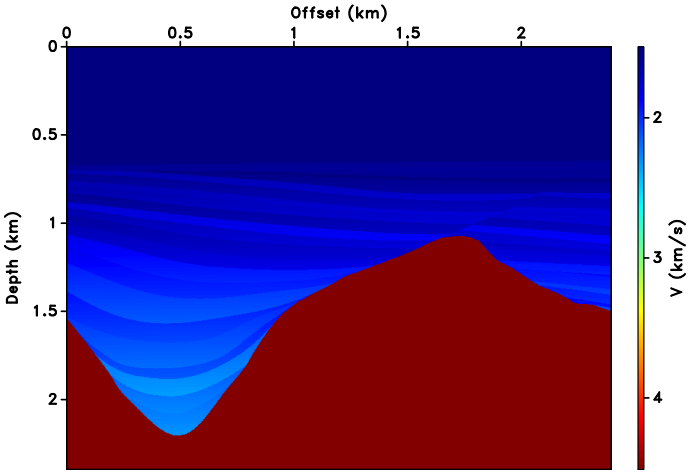


Figure 15: A sub-model of Sigsbee velocity model used for the prediction.

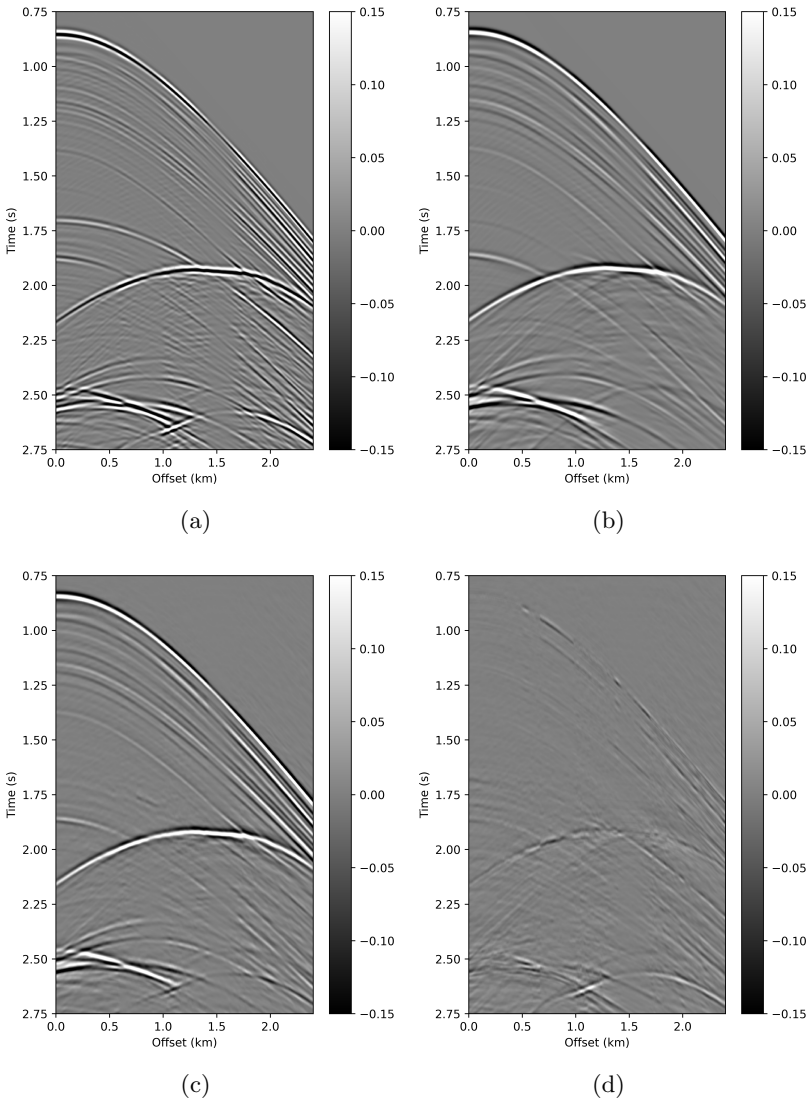


Figure 16: (a) Numerically modeled shot gather with the free-surface multiples (input data) obtained using the velocity model given in Figure 15. (b) Numerically modeled shot gather without the free-surface multiples (output data/ground truth) obtained using the velocity model given in Figure 15. (c) Predicted shot gather without the free-surface multiples using CNNs. (d) Difference between the ground truth data (Figure 16(b)) and the CNN prediction (Figure 16(c))

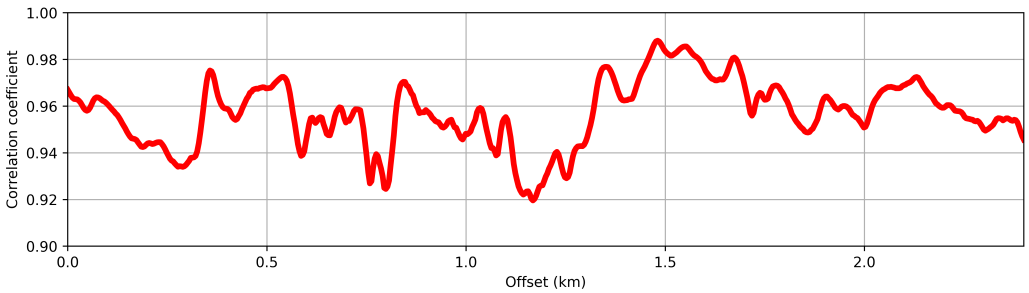
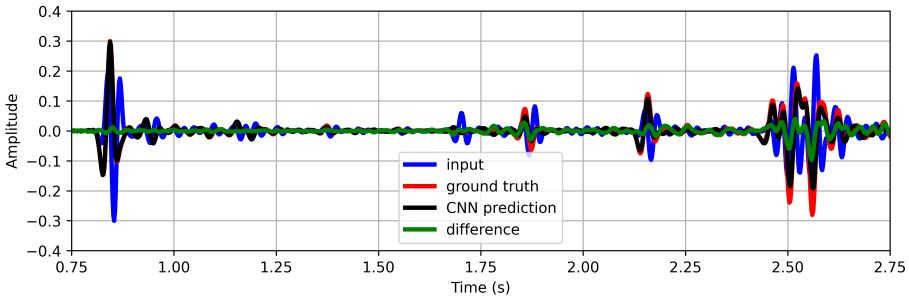
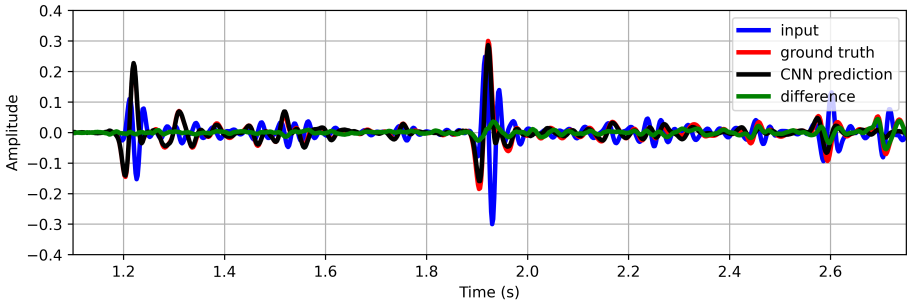


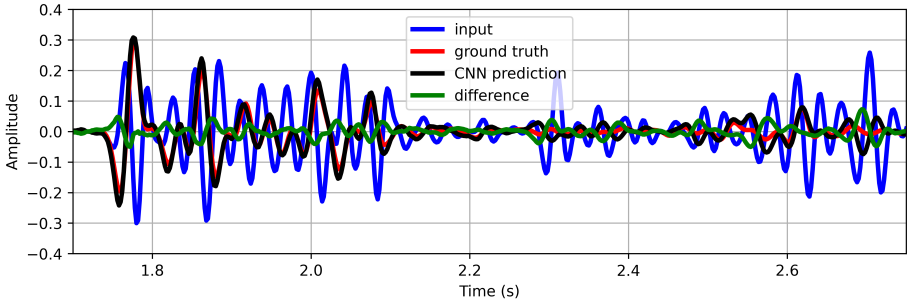
Figure 17: Trace-by-trace calculated correlation coefficient between the ground truth data (Figure 16(b)) and the CNN prediction (Figure 16(c)).



(a)



(b)



(c)

Figure 18: Trace comparison of the input (solid blue line), ground truth (solid red line), CNN prediction (solid black line), and the difference between the ground truth and the CNN prediction (solid green line) data from the zero-, mid-, and far-offsets, respectively, extracted from the data shown in Figure 16. (a) Comparison of the zero-offset traces. (b) Comparison of the mid-offset (offset 1.5 km) traces. (c) Comparison of the far-offset (offset 2.3 km) traces.

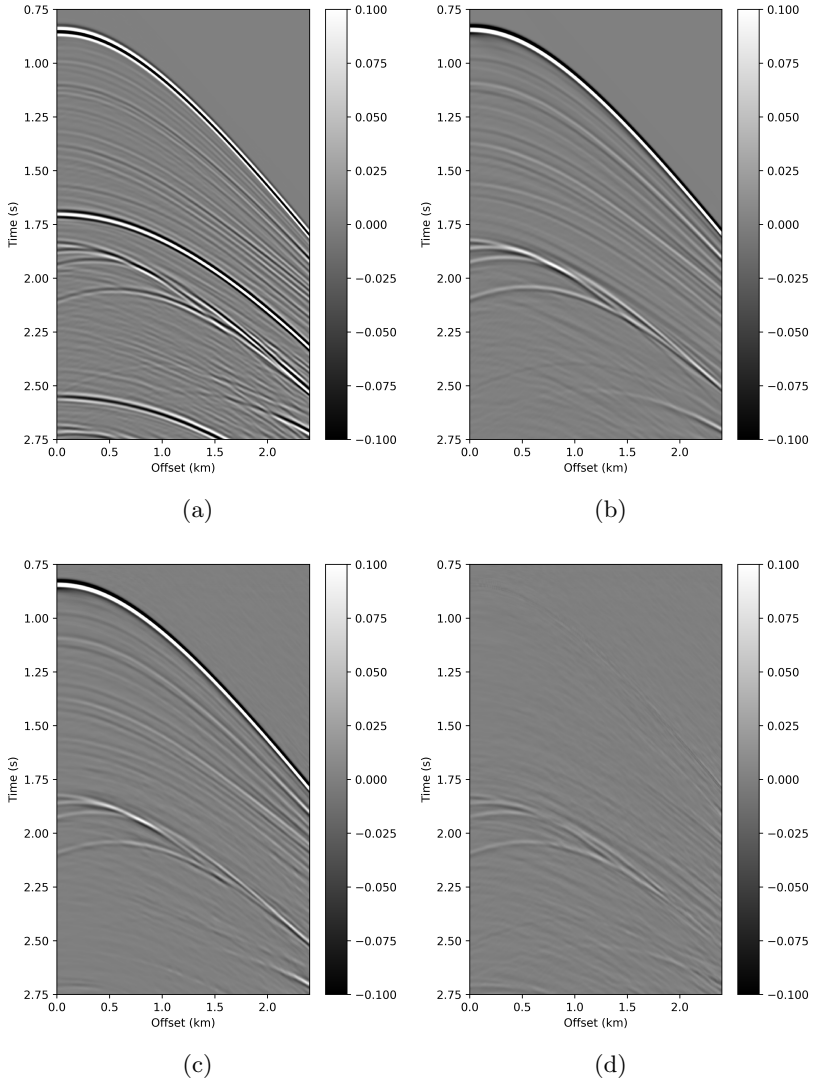


Figure 19: (a) Numerically modeled shot gather with the free-surface multiples (input data) with a reflection coefficient of 0.5 at the sea-bottom obtained using the velocity model given in Figure 11. (b) Numerically modeled shot gather without the free-surface multiples (output data/ground truth) obtained using the velocity model given in Figure 11. (c) Predicted shot gather without the free-surface multiples using CNNs. (d) Difference between the ground truth data (Figure 19(b)) and the CNN prediction (Figure 19(c)).

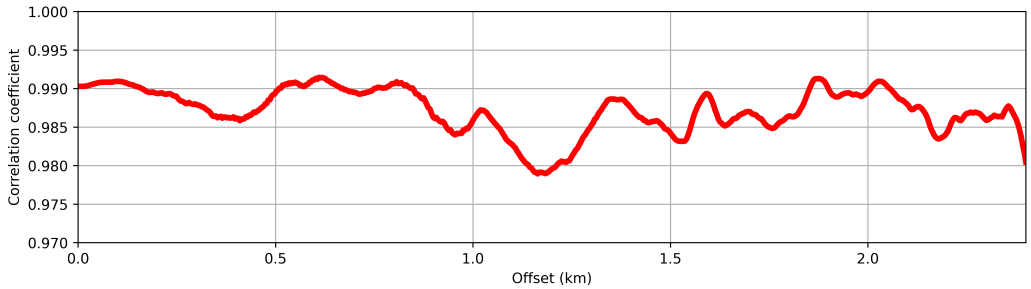
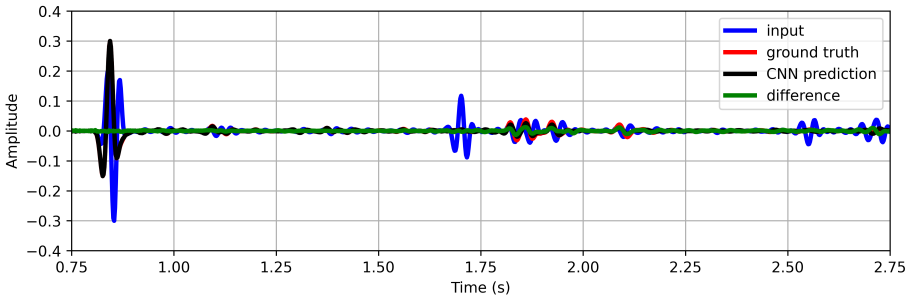
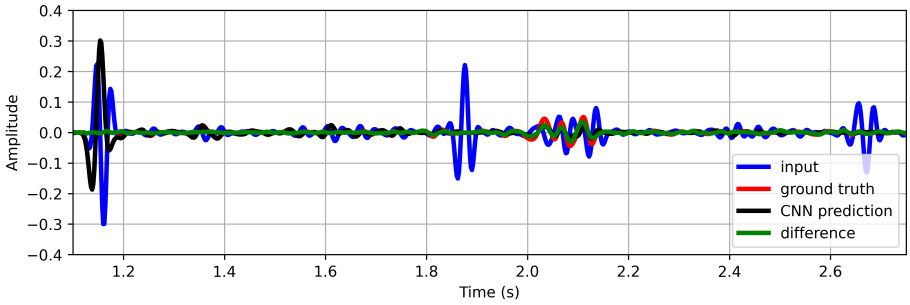


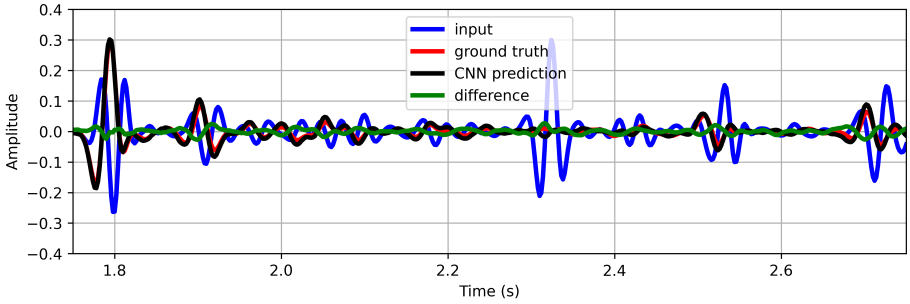
Figure 20: Trace-by-trace calculated correlation coefficient between the ground truth data (Figure 19(b)) and the CNN prediction (Figure 19(c)).



(a)



(b)



(c)

Figure 21: Trace comparison of the input (solid blue line), ground truth (solid red line), CNN prediction (solid black line), and the difference between the ground truth and the CNN prediction (solid green line) data from the zero-, mid-, and far-offsets, respectively, extracted from the data shown in Figure 19. (a) Comparison of the zero-offset traces. (b) Comparison of the mid-offset (offset 1.2 km) traces. (c) Comparison of the far-offset (offset 2.3 km) traces.

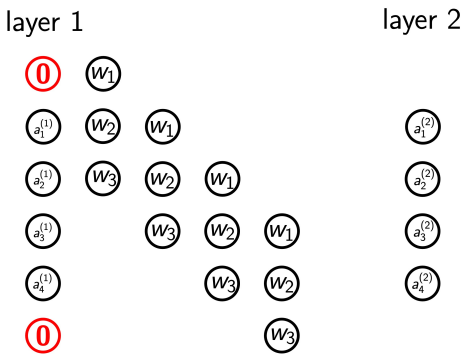


Figure 22: A sketch of a zero-padded 1D convolutional neural network for a two-layer network where the number of output traces for a layer is three.

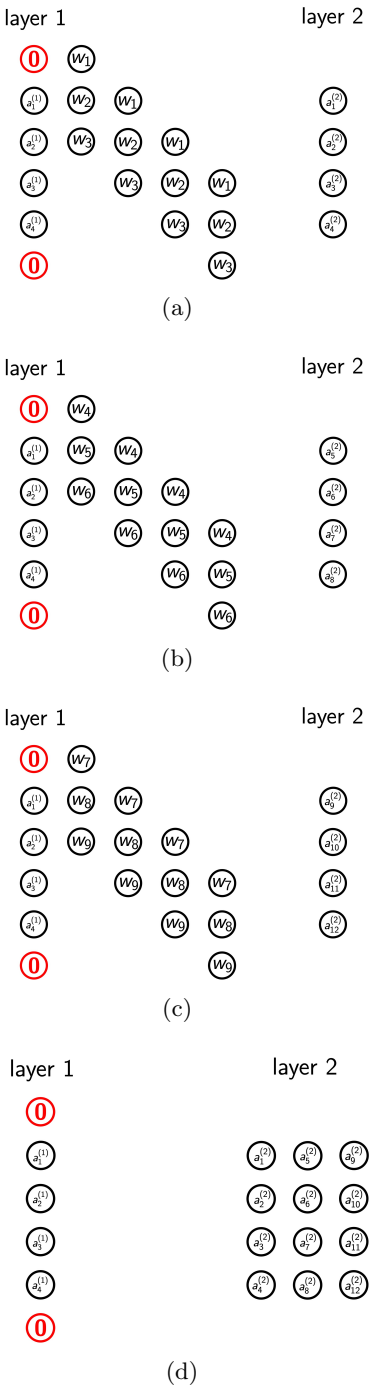


Figure 23: A sketch of a 1D convolutional neural network for a two-layer network along with three three-sample temporal windows. (a) The first filter and its output on *layer 2*. (b) The second filter and its output on *layer 2*. (c) The third filter and its output on *layer 2*. (d) Output of three three-sample windows on *layer 2*.