A convolutional neural network for source range and ocean seabed classification using pressure time-series

David Van Komen, Tracianne B. Neilsen, David P. Knobles, and Mohsen Badiey

Citation: Proc. Mtgs. Acoust. **36**, 070004 (2019); doi: 10.1121/2.0001124
View online: https://doi.org/10.1121/2.0001124
View Table of Contents: https://asa.scitation.org/toc/pma/36/1
Published by the Acoustical Society of America

Volume 36 http://acousticalsociety.org/

177th Meeting of the Acoustical Society of America

Louisville, Kentucky

13-17 May 2019

Underwater Acoustics: Paper 5aUW5

A convolutional neural network for source range and ocean seabed classification using pressure time-series

David Van Komen
Department of Physics & Astronomy, Brigham Young University, Provo, UT, 84602; david.vankomen@byu.edu; david.vankomen@gmail.com

Tracianne B. Neilsen
Brigham Young University, Provo, UT, 84602; tbn@byu.edu

David P. Knobles
KSA, LLC, Austin, TX; dpknobles@kphysics.org

Mohsen Badiey
University of Delaware, Newark, DE; badiey@udel.edu

Neural networks learn features that are useful for classification directly from a source, such as a recorded signal, which removes the need for feature extraction or domain transformations necessary in other machine learning algorithms. To take advantage of these benefits and have a finer temporal resolution, a one-dimensional convolutional neural network is applied to pressure time-series to find source range and ocean environment class from a received signal. The neural network was trained on simulated signals generated in different environments (sandy, muddy, or mixed-layer sediment layers) for several ranges (0.5 to 15 km). We found significant potential in a neural network of this type, given a large amount of varied training samples for the network, to learn important features suitable for range and environment predictions. This type of network provides an alternative for frequency-domain learning and is potentially useful for impulsive sources. Success in the time domain also reduces the computational requirements of conversion to frequency domain and increases the temporal resolution, which might be beneficial for real-time applications.

Published by the Acoustical Society of America

 https://doi.org/10.1121/2.0001124

1. INTRODUCTION

In ocean acoustics research, localizing acoustic sources and estimating the ocean environment present many challenges. Some challenges of these inverse problems include nonlinear relationships between the unknowns, high-dimensional search spaces, and large uncertainty due to ill-conditioning of the inferred parameters. Solutions can be done by using optimization techniques, like matched field processing,[1] but these are often limited to correct or inferred environmental information. Beyond optimization techniques, machine learning and deep learning are increasingly becoming of interest due to their ability to extract features and patterns from data.

Recent efforts have used machine learning to estimate ranges of sources. Steinberg *et al.*[2] introduced the idea of using neural network techniques to localize an acoustic point source in a homogeneous medium using single- and two-layer neural networks. Hougnigan *et al.*[3] used supervised machine learning for real-time range estimation and obtained an average error of 4.3%. Lefort *et al.*[4] and Niu *et al.*[5,6] found that machine learning classifiers outperform traditional methods such as inversion and matched field processing. Others, such as Huang *et al.*[7] and Wang *et al.*[8] have had similar successes with using machine learning techniques to localize sources in the ocean. Machine learning has also been used for classification of underwater targets as done by Fischell *et al.*[9]

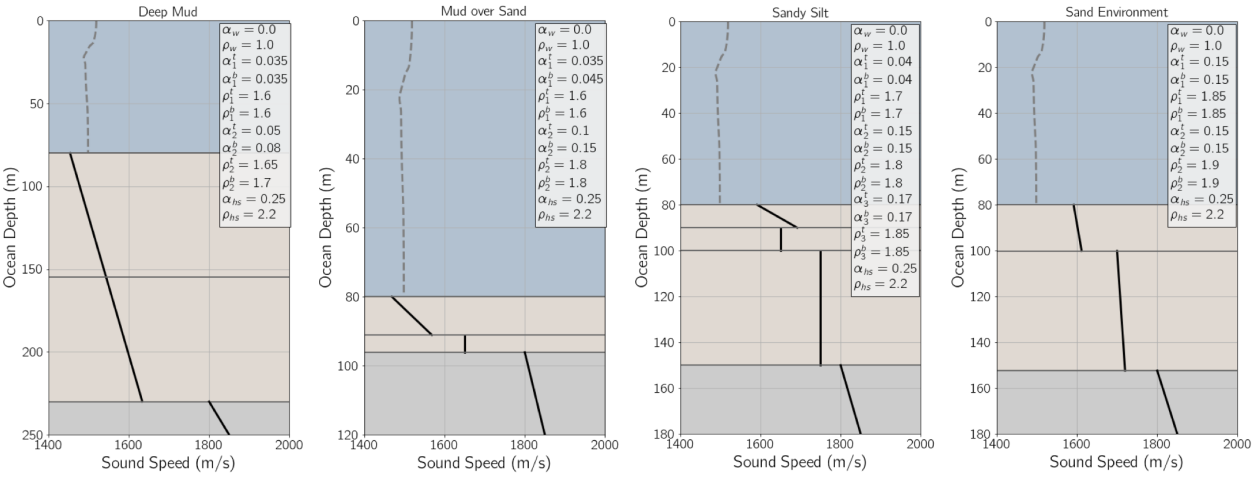
Given the precedence for source localization in the ocean, the next step is determining if these models can also learn environmental information from the data. The work of Piccolo *et al.*[10] estimated seabed parameters, such as sound speed and attenuation, with machine learning on extracted features from signals. In Van Komen *et al.*,[11] an attempt to use feedforward neural networks (FNN) to predict range and seabed type showed that models can learn to predict both simultaneously. The goal of this paper is to expand beyond using a simple feedforward neural network (FNN)[12] for predicting environmental information from full pressure time-series through the use of convolutional neural networks. This paper contains preliminary results on a simulated dataset. Future work will use more sophisticated networks with simulated and real datasets.

2. BACKGROUND

A. UNDERWATER ACOUSTICS AND SUS CHARGES

To generate the data used in this experiment, the signals underwater sound[13] (SUS) charge spectra were simulated through four different environments representing deep mud,[14] mud over sand,[15] sandy silt,[16] and sandy environments.[17] The SUS charge simulations follow the model by Chapman[13] as explained in Sen *et al.*[18] for a 0.034 kg charge in all cases. The parameterization of these four environments is displayed in Figure 1. These figures show a vertical "slice" from the surface of the ocean to the sediment layers that constitute the ocean floor. The water column sound speed profile, typical of a shallow ocean, is shown in the blue, with constant attenuation and density listed in the legend with the subscript w. Each sediment layer l is parameterized by it's thickness h_l in m, compressional sound speed c_l at the top and bottom of the layer in m/s, density ρ_l at the top and bottom in g/cm^3, and compressional attenuation α_l at the top and bottom in dB/λ. The bottom boundary, or half-space, is defined by the compressional sound speed, attenuation and density. These parameters are all fed into the normal mode model to simulate the response of the ocean.

The response of the ocean environment is simulated with ORCA.[19] The ORCA model calculates normal mode functions in range-independent environments by calculating upward- and downward-reflecting plane wave reflections. The mode functions can then be used to generate an impulse response of the ocean at a certain range away from the source. The impulse response is convolved with the simulated source signal at depths $z_s = 4, 18.3,$ and 35 m, to produce the time series signals that comprise this study's dataset.



(a) Deep mud ***(b) Mud over sand*** ***(c) Sandy silt*** ***(d) Sand***

Figure 1: Parameterization of the four environments used in this study. These environments increase in reflectivity and are numbered 1 through 4 from left to right for the use of the machine learning algorithms.

B. MACHINE LEARNING METHODS - CONVOLUTIONAL NEURAL NETWORKS

A feedforward neural network[12] (FNN) is a powerful universal function estimator, but it lacks the ability to detect patterns independent of where they lie in the input space. For example, if the raw pixels of an image were fed directly into a trained FNN, there is no guarantee that a particular pattern near the left side of the image would trigger the same feature response as if it were on the right side of the image. FNNs make predictions from extracted features as they are always ordered vectors with each extracted feature occupying the same position for every sample, but that isn't guaranteed for other input data where they might not be aligned or sorted.

To solve this problem in image recognition, the convolutional neural network (CNN) was developed for grid-like data such as time-series data (one-dimensional grids) or images (two-dimensional grids). It is important to understand that a CNN is simply a neural network that uses at least one convolution layer in place of a general hidden layer seen in an FNN. However, most CNN architectures have fully connected layers that are also present in the FNN. Convolutional layers do feature extraction from the raw input data and then feed those features into the fully connected layers. This removes the need for feature extraction to be done by hand.

In image processing, a CNN takes an input image and slides multiple trained two-dimensional "filters" across the entirety of the image. These learned filters are identifying areas of the image that are of particular interest. However, the power of a CNN comes from stacking multiple convolutional operations in sequence to make a "deep" network. This stacking of convolutional layers allows the network to detect high- and low-level features through training.

To explain the mathematics behind this convolution operation used in CNNs for one-dimensional grids, it is useful to understand discrete one-dimensional convolutions. For the following definition, it must be assumed that time is discretized, and that the data follows the discretization of time, thus t can only take on integer values. To perform this operation $s(t)$, input x convolved with a filter w (where both x and w are

defined only on integer t) can be expressed by

$$\begin{aligned} s(t) = (x * w)(t) &= \int x(a)w(t-a)da \\ &= \sum_{a=-\infty}^{\infty} x(a)w(t-a). \end{aligned} \tag{1}$$

Due to t being defined as an integer, the sum is simplified due to $da = 1$. The filter w, also referred to as the weight vector or the kernel, is dynamically learned through gradient descent which is a process that is beyond the scope of this paper and was automated by the PyTorch[20] Python library. In this particular study, the Adam optimizer[21] was used.

When implementing a CNN, networks will most often use convolution operations that return multiple channels, meaning that there are multiple different weight vectors to be learned. This results in a two-dimensional output after this first layer. When stacking multiple convolutional layers in a network, Equation 1 holds for the first layer, but subsequent layers need a two-dimensional kernel. This is because CNNs combine the features given by the multiple channels at once to learn higher-level features. If the two-dimensional kernel is now K and the new feature map is I, Equation 1 becomes

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i-m, j-n)K(m,n) \tag{2}$$

where K is an $m \times n$ filter where m is the number of channels and n is the length of the desired kernel.

With the definition of the one- and two-dimensional convolution operations, it is important to understand the dimensionality of the outputs at the end of each operation. As a general rule of thumb, when using single-channel data (such as black and white images or a single hydrophone recording, for example), the input number of dimensions is N and all subsequent outputs have $N + 1$ dimensions. However, when using multi-channel data (such as color images coded by red, blue, and green, or multiple hydrophones), the number of dimensions N remains unchanged after using the convolution operation. This is because convolutional neural networks seek to find more intricate features by combining multiple features extracted in previous layers.

To understand how this is implemented in PyTorch, for example, a single-dimensional signal is input into a CNN. To set up the convolution operations, the developer would use the `Conv1d` function included in the package and set $n_{\text{in channels}} = 1$ and $n_{\text{out channels}}$ to the desired number of channels for that layer. The next layer would then include a `Conv1d` function set with $n_{\text{in channels}}$ equal to the previous number of channels and $n_{\text{out channels}}$ to the next desired number of channels. This would continue throughout the entire network for however many convolutional layers the developer desires. As a clarification, the `Conv1d` function is performing one-dimensional convolutions across channels using a two-dimensional weight filter; similarly, the `Conv2d` function for two-dimensional input data performs two-dimensional convolutions with a three-dimensional weight filter. The number in the name of the function refers to the number of dimensions over which the filter is being moved.

In the first and subsequent layers, the output features are also passed through an activation function to introduce non-linear properties into the network. In particular, this paper opted to use the rectified linear unit (ReLU) activation function.[22] During training, batch normalization[23] was also used to accelerate learning by allowing multiple samples to be input at once to take advantage of GPU processing. These equations come from Goodfellow's Deep Learning book[24] and see Chapter 9 for a more in-depth explanation of using and implementing CNNs in machine learning.

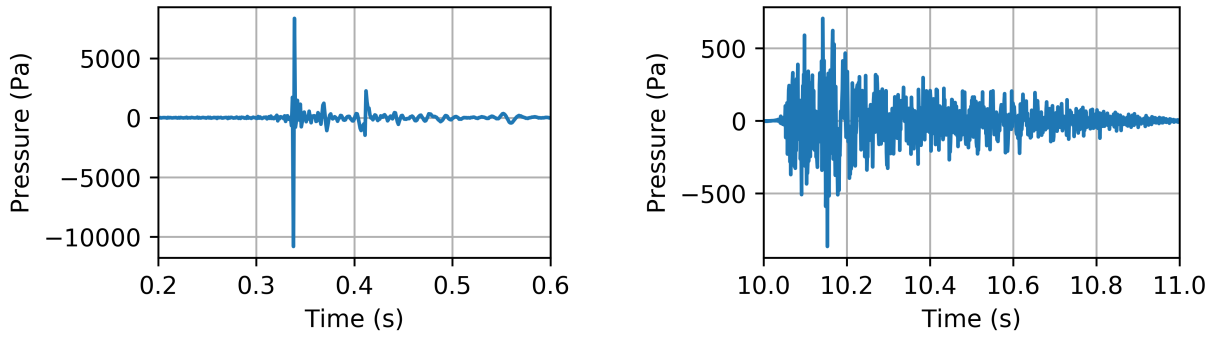
3. METHOD

Predicting seabed type and source range with a CNN requires training data. This section illustrates how the data were generated, how specific features summarizing the data were selected, and how the network was designed for this problem.

A. DATA SIMULATION AND FEATURE EXTRACTION

For this experiment, data were generated with SUS[13] charges in environments simulated by the range-independent ocean model ORCA.[19] The four different environments (muddy, mud over sand, sandy slit, and sand) are shown in Figure 1. The charges were simulated at 30 equally-spaced ranges ($r = 0.5 - 15$ km) away from a receiver. The charges were also simulated at three different depths ($z_s = 4$, 18.3, or 35 m). The environment type, range value, and explosion depth are the values that the network attempts to learn to predict. In the case of regression, the network attempts to predict the actual values independently as a regression problem, though the classification case attempts to predict which unique combination of values the sample belongs to.

The SUS charges were also simulated with 20 different water column profiles sampled from real and typical shallow-water measurements in an ocean 80 m deep to provide variability in the samples. The signals are approximately 14 seconds at a sampling rate of 1,000 Hz making each consist of approximately 14,000 discrete pressures. The 14 seconds was chosen to maintain absolute travel time within the signal which provides extra information about the range. Figure 2 shows two example signals from the dataset.



(a) $r = 0.5$ ***km,*** $z_s = 4$ ***m, deep mud.*** ***(b)*** $r = 15$ ***km,*** $z_s = 35$ ***m, sand.***

Figure 2: Example signals from the dataset. The total length of each is 14 seconds at a sampling rate of 1000 Hz. The two samples shown here illustrate range (r), source depth (z_s), and environment. The different time axes show that absolute arrival time is maintained in this dataset.

Before the data goes into the network, the dataset first needs to be normalized. For this particular case, the data were normalized by the absolute maximum of the entire dataset. This normalization not only places all values between -1 and 1, but it also maintains relative amplitude across the whole dataset. Another method considered is normalizing each sample individually by its own maximum, but in that case the relative amplitude information is lost.

Table 1: Details of the CNN. Layers in the network along with operation type, kernel size, number of channels, stride, and what type of activation function was used at the end of the layer. In the case of the linear operations, the kernel size indicates the number of output nodes. "Conv1D" is an abbreviation for one-dimensional convolution.

Layer	Layer Operation	Kernel Size	Channels	Stride	Activation	Batch Norm
1	Conv1D	16	3	4	ReLU	Y
2	Conv1D	8	9	4	ReLU	Y
3	Conv1D	8	18	4	ReLU	Y
4	Conv1D	8	27	4	ReLU	Y
5	Linear	500	N/A	N/A	ReLU	Y
6	Linear	3 or n_{classes}	N/A	N/A	Linear or Softmax	N

B. CONVOLUTIONAL NEURAL NETWORK TOPOLOGY AND HYPERPARAMETERS

The CNN used to predict environment and source range was built in Python with the PyTorch[20] package. PyTorch* is an open-source deep-learning platform that uses native Python syntax to quickly prototype, train, and test networks of any configuration. PyTorch also automates the required algorithms to perform gradient descent, learning rate scheduler components, and other useful algorithms making it ideal for proof-of-concept and production work. The CNN used consisted of 4 convolutional layers and 2 linear operations. The hyperparameters corresponding to each of the network layers can be found in Table 1.

The network was trained with the Adam optimizer[21] using a learning rate of 0.001 that annealed via a cosine function (a function included in PyTorch[20]) over 1,500 epochs. The loss function chosen was mean squared error loss,[24] which is useful for predicting the true physical values through regression. The learning rate was annealed to allow the network to approach the optimal weights early in training and then refine them later in training. The 7,200 data samples were split into training and testing datasets with a random 80/20 split (5,760 training samples, and 1,440 testing samples randomly divided each training instance). The results of training this network are shown using 1,440 testing samples not used during training.

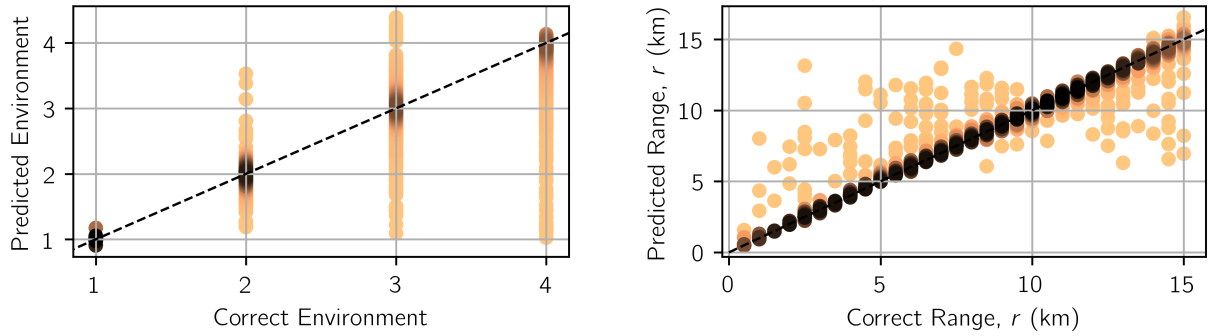
4. RESULTS

The results have been divided into two sections. The first sections shows the performance of a FNN on this dataset of simulated pressure time series. (A previous study with an FNN was limited to 135 data samples.[11]) The second section provides the results for the CNN detailed in Section 3.B on the dataset.

A. FEEDFORWARD NEURAL NETWORK RESULTS

To provide a baseline result, the dataset was input into a FNN. This FNN consisted of two hidden layers with 5,000 nodes each, was trained across over 300 epochs, and predicted a number between 1 and 4 for a seabed type, and value in km for the source-reciever range of the signal. Figure 3 shows the predictions the trained network made on environment class and source range independently on the validation data. The colors in Figure 3 are used to show the density of the predictions since there are over one thousand points on each plot. The FNN was asked to only predict the environment number and the source range to better match the previous study on FNNs,[11] where more information about setting up a FNN can be found.

*More information about how to use PyTorch can be found on their website at https://pytorch.org/.



(a) Predicted vs correct environment.

(b) Prediction vs correct range.

Figure 3: Prediction results from the FNN. There are 1,440 different points on these plots, so the colormap scale shows the density of points (the darker the color, the denser the scatter plot). When the environment predictions (Figure 3a) are rounded to the nearest integer, an 85% accuracy is achieved. The source range predictions (Figure 3b) have a RMSE of 1.19 km when compared to the true range. The dashed line shows the values that would be obtained if the network made an exactly correct prediction.

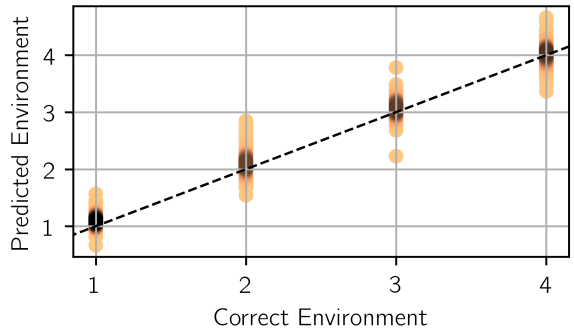
To determine the accuracy of the environment class, the predicted environment value was rounded to the nearest integer and compared against the true environment. With this rounding, the network was 85% accurate on environment. The network correctly identifies the first environment, but as soon as more reflectivity is added, the network struggles, especially with the sandy silt and sandy seabeds. However, the dark color near the correct answer indicates a high density of points near there, so only a small number of these samples are actually incorrect as indicated by the 85% accuracy. The source-receiver range predictions had a root mean squared error (RMSE) of 1.19 km when compared to the true range. The FNN tends to over-predict the range until around 8 km and then it begins to under-predict the range. However, the FNN still gets close to correct range on most of the training samples.

If this FNN network was instead configured for classification, the FNN was 71.2% accurate on the 120 unique range-seabed classes. These results are significantly better than the results that used pressure time-series instead of extracted features found in a previous study using FNNs.[11] This improvement could be due to the larger training dataset or the inclusion of absolute travel time in the signals.

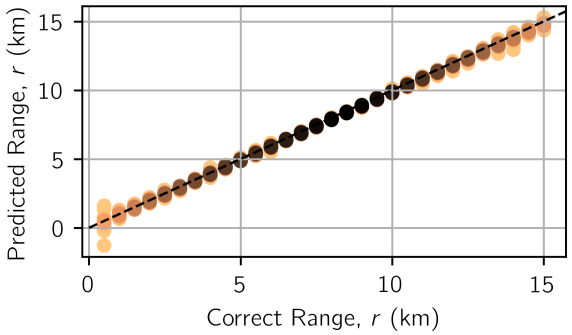
B. CONVOLUTIONAL NEURAL NETWORK RESULTS

The CNN was trained on the same dataset but was designed and trained to learn environment number, source range r, and source depth z_s. Figure 4 shows the results of the CNN on the same size of validation dataset from a different random split. A visual indicator that the network has learned well is how clustered the different values are and how dark the points get near the correct answer line.

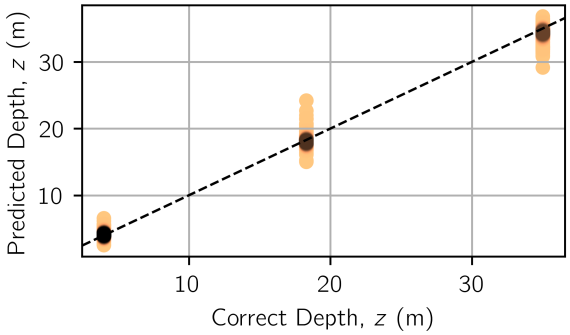
These figures show interesting trends when compared to the FNN. First, in Fig. 4, the predictions all appear closer to the true value across all samples and all predictions. As for the seabed environment prediction (as seen in Fig. 4a), there are several predictions that are outside the dense cluster, but the majority are correctly predicted. The error in range prediction (as seen in Fig. 4b) increases towards the min and max ranges, but are still relatively close to the correct values. This increase is likely due to the fact that there are not training data points beyond these extremes to help the CNN distinguish these ranges. In comparison with the FNN results, which had some wildly inaccurate predictions, the CNN predictions are mostly correct or incorrect by only a small margin. The source depth predictions (as seen in Fig. 4c) show



(a) Predicted vs correct environment.



(b) Predicted vs correct range.



(c) Predicted vs correct source depth.

Figure 4: Prediction results from the CNN. There are 1,440 different points on these plots, so the colormap scale is used to show density of points (the darker the color, the denser the scatter plot). When the environment predictions (Figure 4a) are rounded to the nearest integer, a 98.96% accuracy is achieved. The source range predictions (Figure 4b) has a RMSE of 0.1798 km when compared to the true range. The source depth predictions (Figure 4c) has a RMSE of 0.7633 m. The dashed line shows the values that would be obtained if the network made an exactly correct prediction.

a distinct grouping around the correct source depth.

Comparison of the accuracy confirms the superiority of the CNN to the FNN. The RSME for the CNN range predictions is 0.1798 km—an order of magnitude better than the predictions of the FNN. The accuracy of the environment prediction is 97%, which is 13% higher than that of the FNN. The CNN also predicts a third label, the source depth, with an RMSE of 0.7633 m. The CNN performed much better than the FNN even when predicting an additional label.

The results described come from the CNN structured for regression, but the final output layer can be modified for classification. Beforehand, each sample must be labeled into a class representing its combination of source-receiver range, source depth, and environment. This creates 360 unique classes for the network to learn. The same 80/20 training/testing data split was used. The CNN was 97% accurate in classifying the 1440 testing samples, which is 26% higher than the FNN on triple the number of classes.

5. CONCLUSION

This paper has shown that CNNs have the potential to learn source range and environmental classification from time-series waveforms. The CNN on the waveforms performed better than an FNN applied to the same set of data. This improvement implies a significant advantage to using convolutions on the grid-like structure of pressure time-series data. This study also lays the ground work for developing deeper and more complex networks, such as U-net CNNs[25] and recurrent neural networks (such as long short-term memory networks[26]), which have also shown success in image and speech recognition. Future work seeks to expand on these results by increasing the amount of data, increasing the number of environments used, application of noise to training and testing data, and applying the networks to real measured data.

ACKNOWLEDGMENTS

This work was funded in part by the Office of Naval Research, SBIR Grant #N68335-18-C-0806 and by Office of Naval Research contract number N00014-19-C-20001.

REFERENCES

[1] A. Tolstoy, *Matched Field Processing for Underwater Acoustics* (World Scientific, 1993).

[2] B. Z. Steinberg, M. J. Beran, S. H. Chin, and J. H. Howard Jr, "A neural network approach to source localization," J. Acoust. Soc. Am. **90**(4), 2081–2090 (1991).

[3] L. Hougnigan, P. Safari, C. Nadeu, M. van der Schaar, and M. Andr, "A novel approach to real-time range estimation of underwater acoustic sources using supervised machine learning," in *OCEANS 2017-Aberdeen*, IEEE (2017), pp. 1–5.

[4] R. Lefort, G. Real, and A. Drmeau, "Direct regressions for underwater acoustic source localization in fluctuating oceans," J. Appl. Acoust. **116**, 303–310 (2017).

[5] H. Niu, E. Ozanich, and P. Gerstoft, "Ship localization in Santa Barbara Channel using machine learning classifiers," J. Acoust. Soc. Am. **142**(5), EL455–EL460 (2017).

[6] H. Niu, E. Reeves, and P. Gerstoft, "Source localization in an ocean waveguide using supervised machine learning," J. Acous. Soc. Am. **142**(3), 1176–1188 (2017).

[7] Z. Huang, J. Xu, Z. Gong, H. Wang, and Y. Yan, "Source localization using deep neural networks in a shallow water environment," J. Acoust. Soc. Am. **143**(5), 2922–2932 (2018).

[8] Y. Wang and H. Peng, "Underwater acoustic source localization using generalized regression neural network," J. Acoust. Soc. Am. **143**(4), 2321–2331 (2018).

[9] E. M. Fischell and H. Schmidt, "Classification of underwater targets from autonomous underwater vehicle sampled bistatic acoustic scattered fields," J. Acoust. Soc. Am. **138**(6), 3773–3784 (2015).

[10] J. Piccolo, G. Haramuniz, and Z.-H. Michalopoulou, "Geoacoustic inversion with generalized additive models," J. Acoust. Soc. Am. **145**(6), EL463–EL468 (2019).

[11] D. F. Van Komen, T. B. Neilsen, D. P. Knobles, and M. Badiey, "A feedforward neural network for source range and ocean seabed classification using time-domain features," Proc. Meet. Acoust. **36**(1), 070003 (2019).

[12] T. D. Sanger, "Optimal unsupervised learning in a single-layer linear feedforward neural network," Neural Netw. **2**(6), 459–473 (1989).

[13] N. R. Chapman, "Source levels of shallow explosive charges," J. Acoust. Soc. Am. **84**(2), 697–702 (1988).

[14] D. P. Knobles, R. A. Koch, L. A. Thompson, K. C. Focke, and P. E. Eisman, "Broadband sound propagation in shallow water and geoacoustic inversion," J. Acoust. Soc. Am. **113**(1), 205–222 (2003).

[15] D. P. Knobles, P. S. Wilson, J. A. Goff, L. Wan, M. J. Buckingham, J. D. Chaytor, and M. Badiey, "Maximum entropy derived statistics of sound-speed structure in a fine-grained sediment inferred from sparse broadband acoustic measurements on the New England Continental Shelf," IEEE J. of Ocean. Eng. 1–13 (2019).

[16] G. R. Potty, J. H. Miller, and J. F. Lynch, "Inversion for sediment geoacoustic properties at the New England Bight," J. Acoust. Soc. Am. **114**(4), 1874–1887 (2003).

[17] J.-X. Zhou, X.-Z. Zhang, and D. P. Knobles, "Low-frequency geoacoustic model for the effective properties of sandy seabottoms," J. Acoust. Soc. Am. **125**(5), 2847–2866 (2009).

[18] M. K. Sen, L. N. Frazer, S. Mallick, and N. R. Chapman, "Analysis of multipath sound propagation in the ocean near 49° N, 128° W," J. Acoust. Soc. Am. **83**(2), 588–597 (1988).

[19] E. K. Westwood, C. T. Tindle, and N. R. Chapman, "A normal mode model for acoustoelastic ocean environments," J. Acoust. Soc. Am. **100**(6), 3631–3645 (1996).

[20] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch," in *NIPS-W* (2017).

[21] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv preprint arXiv:1412.6980 (2014).

[22] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann Machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)* (2010), pp. 807–814.

[23] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," arXiv preprint arXiv:.03167 (2015).

[24] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning* (MIT press, 2016).

[25] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*, Springer (2015), pp. 234–241.

[26] S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural computation **9**(8), 1735–1780 (1997).