Proceedings of Meetings on Acoustics

SEPTEMBER 03 2021

Three-dimensional convolutional neural networks for target classification with volumetric sonar data FREE

David P. Williams ; Daniel C. Brown

Proc. Mtgs. Acoust. 44, 070005 (2021)
https://doi.org/10.1121/2.0001453

Volume 44 http://acousticalsociety.org/

6th Underwater Acoustics Conference & Exhibition

20-25 June 2021

Underwater Acoustics: Underwater Unexploded Ordnance (UXO) Detection and Remediation

Three-dimensional convolutional neural networks for target classification with volumetric sonar data

David P. Williams and Daniel C. Brown
Applied Research Laboratory, The Pennsylvania State University, State College, PA; dzw5587@psu.edu; dcb19@psu.edu

Efficient three-dimensional (3-d) convolutional neural networks (CNNs) are designed and trained for an underwater target classification task with volumetric synthetic aperture sonar (SAS) imagery. (The third dimension of the data represents depth into the sediment, thereby enabling the consideration of *buried* underwater objects.) The use of tiny networks containing relatively few parameters makes training with enormous input data volumes feasible even with modest computational power and limited computer memory. The promise of the approach is demonstrated for both buried and proud man-made objects present in real, measured SAS data cubes collected at aquatic sites by an experimental volumetric sonar system, called the Sediment Volume Search Sonar (SVSS). The classification performance of each 3-d CNN exhibits marked improvement over a prescreening detection algorithm alone, and the utility of an ensemble approach is also quantified. An analysis of the effective functionality of the learned networks is provided, with this also accompanied by figures showing example trained filters as well as intermediate representations of a data volume containing unexploded ordnance (UXO). The predictions of the 3-d CNN classifiers can provide valuable guidance for the efficient allocation of resources during real-world UXO remediation operations.

Published by the Acoustical Society of America

 https://doi.org/10.1121/2.0001453

1. INTRODUCTION

An unfortunate legacy of former military activities is the contamination of aquatic environments with unexploded ordnance (UXO). In shallow water, proud and buried munitions pose a particular threat to both humans and the environment, so remediation is necessary. To address this pressing issue, several low-frequency sonar systems – importantly, on mobile platforms – have recently been developed.[1–3] These downward-looking synthetic aperture sonar (SAS) systems, designed to achieve sediment penetration, provide high-resolution three-dimensional (3-d) *volumetric* imagery below the seafloor, making large-scale buried object detection newly feasible. (Other *side-looking* low-frequency sonars[4,5] generate only 2-d imagery.)

With the introduction of this new sensor modality, there is now a need for automated detection and classification algorithms that can efficiently process enormous 3-d images to rapidly flag suspicious objects for closer inspection during remediation efforts. Relying on humans to visually assess these new data products is both inherently challenging and inefficient. Our recent work[6] presented methods to address the detection stage, while here we propose a novel approach for the follow-on classification stage.

Specifically, the main contribution of this work is the development of 3-d convolutional neural networks (CNNs) for volumetric SAS imagery. A CNN-based classification approach is particularly apropos for this data modality because hand-crafting features is challenging, and CNNs effectively obviate this process. Despite the enormous size of the 3-d data involved, it is shown how careful architecture design can make this proposed approach both feasible and successful. The method is general in the sense that it can be employed for wide classes of objects of interest, but here it is presented in the context of underwater UXO. The proposed CNNs fill a critical capability gap that makes the use of this new class of sensors feasible for real-world UXO remediation operations.

The remainder of this paper is organized as follows. Sec. 2 introduces the volumetric sonar imagery for which novel 3-d CNNs described in Sec. 3 are developed. Sec. 4 presents results of the method on real, measured data for buried and proud man-made targets. Concluding remarks are made in Sec. 5. Two appendices show example learned CNN filters and intermediate responses for one input data cube.

2. VOLUMETRIC SONAR DATA

The Sediment Volume Search Sonar (SVSS)[2] is a recently developed low-frequency sonar system designed to address the UXO remediation problem in shallow water environments (*i.e.,* depths less than 5 m). The experimental system features multiple transmitters and a 2-d receiver array that collectively enable the production of 3-d SAS imagery to facilitate the detection of proud and buried objects. The frequency band of operation is approximately 20-35 kHz. A time-domain back-projection beamformer[7] is used to transform the raw sonar time-series returns into 3-d volumetric imagery comprising voxels that span 2 cm in each dimension. Throughout this paper, the sizes of data cubes and CNN filters will be given in terms of voxels as $x \times y \times z$, where x, y, and z represent the cross-track, along-track, and depth dimensions, respectively.

The challenge of visualizing a 3-d data cube often leads to the use of a 2-d maximum intensity projection (MIP), which collapses the imagery along one of its principal axes by retaining the highest intensity voxels along that axis.[8] A typical data cube from the SVSS system after a normalization algorithm[6] was applied is shown in Fig. 1 as a set of three 2-d MIPs in a common reference frame. Localized alarm data cubes (displayed as MIPs) extracted from Fig. 1 of four targets are shown in Fig. 2, along with corresponding object photographs taken during installation. The 3-d alarm data cubes like in Fig. 2(a)-(d) are what would be the inputs to the CNNs in this work.

For the results reported in this work, the SVSS system was used to collect data at two distinct locations in the Fosters Joseph Sayers Reservoir in Pennsylvania. At these sites, there existed an upper layer of

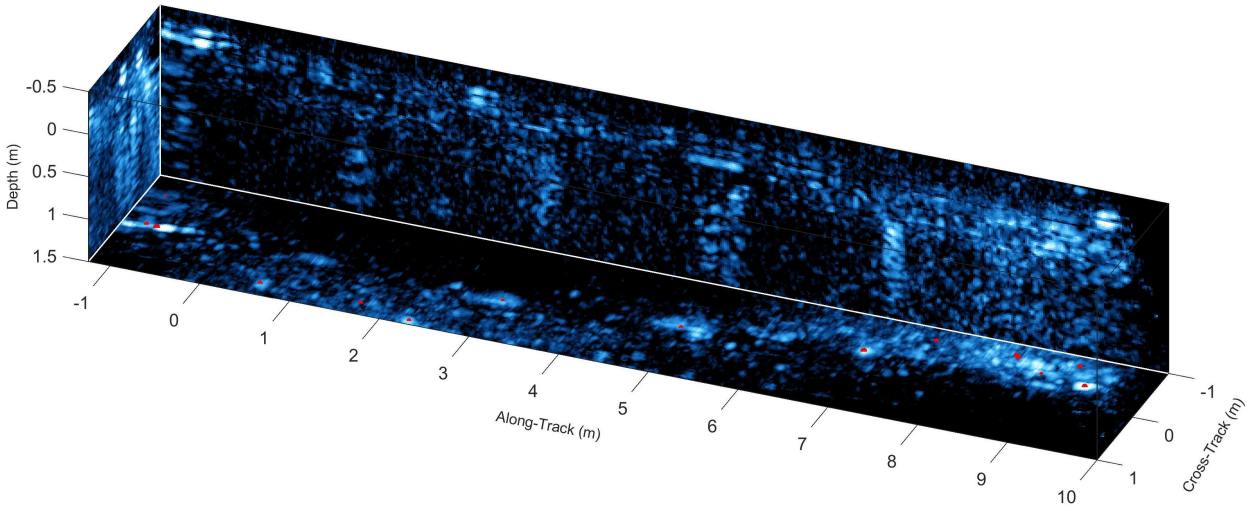


Figure 1: An example (normalized) SVSS volumetric scene image, from site* A*, displayed as a trio of MIPs. Detections passed on to the classification stage are marked on the depth MIPs with red dots.

approximately 8 cm of silt atop a clay base; site "A" had a 1.3 m water depth, while site "B" had a 3.0 m water depth.

The sites were in a reservoir that could be drained to facilitate target emplacement. So prior to data collection, various man-made objects were deployed, including aluminum cylinders, steel pipes, steel shot puts, concrete blocks, and an assortment of munitions with diameters ranging from 81 mm to 155 mm. Some objects were placed proud on the sediment, while others were buried to various depths (up to 33 cm) below the water-sediment interface. Data collections at Sayers in 2019 took place in June, August, early November, and late November. The raw sonar data collected at sites A and B were processed into a total of 521 and 513 volumetric SAS "scene" images (as in Fig. 1), respectively.

A detection algorithm[6] that operates on 3-d imagery was applied to all of this scene-level volumetric data collected at the two sites. The resulting data cubes of alarms from site B were used as training data for the CNNs; the data cubes of alarms from site A were treated as the test set. For the binary-classification experiments in this work, all man-made objects were considered targets, while all other alarms were labeled as belonging to the clutter class. The details of the SVSS data sets after the detection stage are summarized in Table 1.

Table 1: Summary of SVSS sonar data sets after the detection stage

Location	Data Set Usage	Seafloor Area (m^2)	Sediment Volume (m^3)	Number of Clutter	Number of Targets
Site B	Training	15390	23392.8	11029	550
Site A	Test	15630	23757.6	6074	671

3. 3-D CONVOLUTIONAL NEURAL NETWORKS

A CNN is a sophisticated classification algorithm that customarily ingests an image as input and produces scalar outputs corresponding to the probabilities of belonging to each class under consideration (*e.g.,* target and clutter). Within these bookends, the architecture of a CNN consists of a sequence of *layers*, each

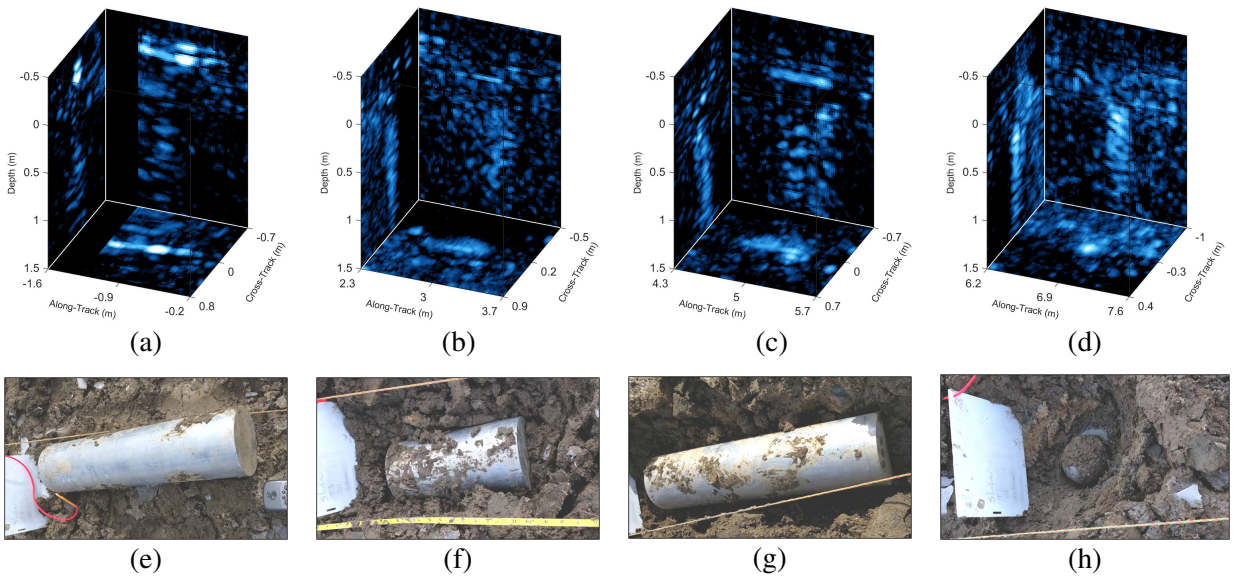


Figure 2: (a)-(d) SVSS alarm cubes (each displayed as a trio of MIPs) of four targets extracted from Fig. 1, and (e)-(h) photographs of the objects during installation (pre-burial). The objects are: (a) 4:1 solid aluminum cylinder proud, (b) 2:1 solid aluminum cylinder buried 5 cm, (c) 4:1 solid aluminum cylinder buried 3 cm, (d) 10.2 cm diameter steel shot put buried 19 cm; the cylinders have 15.2 cm diameters.

performing a specific mathematical operation, arranged such that the output of one layer is the input to the subsequent layer. This nested functional structure – in conjunction with *nonlinear* functions – enables highly complex decision surfaces. In turn, this is the source of a CNN's rich representational capacity.

In this work, we develop *3-d* CNNs in which the input data is a 3-d data cube, rather than a 2-d image. The general architecture design largely follows previous work,[9] but extends it to three dimensions. More specifically, we develop eight CNNs that share a common architecture of alternating convolutional *blocks* (comprising one or more convolutional layers) and pooling layers. The input to a CNN is assumed to be a $1.22\text{ m} \times 1.22\text{ m} \times 2.02\text{ m}$ volumetric SAS image, which corresponds to a $61 \times 61 \times 101$ voxel data cube, since each voxel spans 2.0 cm in each dimension. Thus, a single SAS data cube input to any given network contains approximately 3.75×10^5 voxels. The outputs of the CNN's final layer are the (softmax) probabilities of a 3-d data cube belonging to each class (target or clutter).

Specific details about the designed CNNs are provided in Table 2; the information provided is sufficient for recreating the networks exactly. Here, brackets are used to convey the concept of convolutional *blocks*, in which there are multiple convolutional layers in between pooling layers. (The ith set of brackets contains the information about the convolutional layers in the ith block. For example, the first convolutional block of CNN F comprises a convolutional layer with $7 \times 7 \times 12$ filters, followed by a second convolutional layer with $5 \times 5 \times 7$ filters.) The convolutional block construct allows deeper networks, and thus greater complexity, without a proportional increase in the number of parameters to learn. Despite the enormous size of the input data cubes, the use of extremely small networks allows the CNNs to be trained with modest computational resources, and vitally, avoids computer-memory constraint issues.

Table 2: Architectures of 3-d CNNs trained

CNN Label	CNN Depth	Conv. Blocks	Conv. Layers Per Conv. Block	Filters Per Conv. Layer	Filter Sizes (Voxels) $[x \times y \times z]$	Pooling Factors $[x \times y \times z]$	Number of Parameters
A	2	2	1	4	$[6 \times 6 \times 6]\,[4 \times 4 \times 5]$	$[8 \times 8 \times 12]\,[4 \times 4 \times 4]$	2157
B	2	2	1	4	$[8 \times 8 \times 12]\,[6 \times 6 \times 7]$	$[6 \times 6 \times 9]\,[4 \times 4 \times 4]$	7117
C	3	3	1	4	$[6 \times 6 \times 6]\,[5 \times 5 \times 5]\,[4 \times 4 \times 5]$	$[4 \times 4 \times 6]\,[2 \times 2 \times 2]\,[2 \times 2 \times 2]$	4161
D	3	3	1	4	$[11 \times 11 \times 12]\,[8 \times 8 \times 7]\,[4 \times 4 \times 5]$	$[3 \times 3 \times 5]\,[2 \times 2 \times 2]\,[2 \times 2 \times 2]$	14273
E	6	3	2	4	$\begin{bmatrix} 5 \times 5 \times 6 \\ 4 \times 4 \times 5 \end{bmatrix} \begin{bmatrix} 5 \times 5 \times 6 \\ 4 \times 4 \times 5 \end{bmatrix} \begin{bmatrix} 5 \times 5 \times 3 \\ 4 \times 4 \times 3 \end{bmatrix}$	$[2 \times 2 \times 4]\,[2 \times 2 \times 2]\,[3 \times 3 \times 3]$	7557
F	6	3	2	4	$\begin{bmatrix} 7 \times 7 \times 12 \\ 5 \times 5 \times 7 \end{bmatrix} \begin{bmatrix} 5 \times 5 \times 8 \\ 4 \times 4 \times 5 \end{bmatrix} \begin{bmatrix} 3 \times 3 \times 3 \\ 3 \times 3 \times 3 \end{bmatrix}$	$[3 \times 3 \times 4]\,[2 \times 2 \times 2]\,[1 \times 1 \times 1]$	10525
G	9	3	3	4	$\begin{bmatrix} 4 \times 4 \times 5 \\ 3 \times 3 \times 5 \\ 3 \times 3 \times 4 \end{bmatrix} \begin{bmatrix} 4 \times 4 \times 4 \\ 3 \times 3 \times 4 \\ 3 \times 3 \times 3 \end{bmatrix} \begin{bmatrix} 4 \times 4 \times 4 \\ 4 \times 4 \times 4 \\ 3 \times 3 \times 4 \end{bmatrix}$	$[2 \times 2 \times 3]\,[2 \times 2 \times 2]\,[2 \times 2 \times 2]$	6313
H	12	3	4	4	$\begin{bmatrix} 4 \times 4 \times 3 \\ 3 \times 3 \times 3 \\ 3 \times 3 \times 3 \\ 3 \times 3 \times 3 \end{bmatrix} \begin{bmatrix} 3 \times 3 \times 4 \\ 3 \times 3 \times 3 \\ 3 \times 3 \times 3 \\ 3 \times 3 \times 3 \end{bmatrix} \begin{bmatrix} 3 \times 3 \times 3 \\ 3 \times 3 \times 3 \\ 3 \times 3 \times 3 \\ 3 \times 3 \times 3 \end{bmatrix}$	$[2 \times 2 \times 3]\,[2 \times 2 \times 2]\,[1 \times 1 \times 3]$	5141

Each CNN contains either 2 or 3 convolutional blocks; each block contains a specific number of convolutional layers (equal to the number of rows in Table 2's filter-size columns). A given filter always has the same number of voxels in the x and y dimensions (*i.e.,* it is square), and only 4 filters are used in each convolutional layer. All convolutions use a stride of 1, and padding is not used (*i.e.,* in TensorFlow-speak, the padding option is set to '`valid`'). ReLU activations are used after each convolutional layer, while a softmax activation is used at the output. All pooling layers use average pooling, rather than max pooling. The design of the architecture (and specifically the final pooling layer) ensures that the dense layer always contains 4 nodes. The capacities of the CNNs are intentionally kept so low because the amount of training data available is limited, but also because larger networks would be plagued by computer-memory problems. Collectively, the eight CNNs have only 57244 parameters to learn, which is still several orders of magnitude lower than traditional optical-image CNNs as well as the custom SAS-image CNNs that have been considered in the literature.

At a high-level, the general CNN architecture employed here is not so unusual. However, a few subtle, but key, design choices are a stark break from convention. The most striking deviation is the use of only 4 filters per convolutional layer; almost all CNNs in the literature use many dozens or hundreds or thousands of filters per convolutional layer. Importantly, this self-imposed constraint makes interpretability of the filters and intermediate representations feasible. Additionally, the use of only a *single* dense layer before the output layer is also somewhat unusual. The third uncommon choice is the decision to reduce (via pooling) the size of each feature map in the final convolutional layer to a $1 \times 1 \times 1$ output (*i.e.,* a scalar). Because of this, and the small number of filters, each CNN's dense layer has only 4 nodes. That is, the networks are effectively forced to make predictions from only 4 "features." Although we severely constrain the number of parameters in the models, importantly we do not restrict CNN *depth*. Finally, the retention of a considerable amount of data in the z dimension (*i.e.,* into the sediment) for each input cube enables the CNN to possibly exploit late-arriving elastic scattering phenomena. These clues would likely be inaccessible if 2-d projections (like MIPs) of the data were instead used to train a standard 2-d CNN.

A. CNN TRAINING

CNN training was performed in Python with the TensorFlow[10] software library. Training used an RMSprop optimizer with a learning rate of $\eta = 0.001$, in conjunction with a binary-cross-entropy loss function. A batch size of $B = 64$ was used, with equal numbers selected from each class to combat the severe class-imbalance of the training data. Data augmentation that respected the geometry of the volumetric sonar data-collection procedure was employed during training; this meant a random cross-track translation $i_{tx} \in [-0.1 \text{ m}, 0.1 \text{ m}]$, along-track translation $i_{ty} \in [-0.1 \text{ m}, 0.1 \text{ m}]$, cross-track reflection $i_{rx} \in \{\pm 1\}$, and along-track reflection $i_{ry} \in \{\pm 1\}$ was applied, "on-the-fly," to each SAS image cube selected for the batch. (To permit these translation operations, the data cubes that comprise the training (and test) sets are slightly larger than the size of the data cubes required as input to the CNNs.) No translation or reflection was performed in depth (*i.e.,* the z dimension). It is worth reiterating that fully-populated 3-d volumetric data cubes – and not 2-d MIPs – are used as input to the CNNs.

Each CNN was allowed to train for 200 epochs, where one epoch was defined as a set of 1000 batches. Each batch was formed by randomly selecting image cubes from the full set of training data. (Thus, every 100 epochs during the CNN training phase, each of the 550 training-set target cubes is seen about 5818 times and each of the 11029 training-set clutter cubes is seen about 290 times, on average.) A validation set, common to all CNNs, was created by randomly selecting 50 targets and 450 clutter from the larger training set. For a given CNN, the epoch for which the model achieved the maximum AUC on the validation set was used to select the final model to evaluate the test sets. Based on this criterion, the number of training epochs actually used to train each CNN is shown in Table 3. For reference, training time for these CNNs is about 2 seconds per batch with a single GeForce GTX 1080 GPU. Thus, based on the actual number of training

epochs needed, the training time for a single CNN is on the order of only 10 to 100 hours.

Table 3: Training epoch at which validation set AUC was maximized

CNN	Training Epoch
A	163
B	145
C	56
D	56
E	53
F	17
G	26
H	56

Appendix A shows the intermediate responses at each layer of one CNN for a single interesting data cube containing UXO. The convolutional filters learned by one of the CNNs are shown in Appendix B.

4. EXPERIMENTAL RESULTS

A. PRELIMINARIES

Our main interest in this paper is to examine the ability of the CNN-based approach to successfully perform *classification*. Therefore, when presenting receiver operating characteristic (ROC) curves, the experiments assume that all targets were successfully flagged in the detection stage. Thus, the maximum possible area under the ROC curve (AUC),[11] a scalar summary measure of performance, is always unity. (A perfect classifier would have an AUC of unity.) To obtain the overall probability of both successful detection and correct classification, the vertical axis of the ROC-like curves would need to be scaled by the total number of targets present in the scene imagery.

B. CLASSIFICATION RESULTS

First, we examine the benefit of making CNN predictions based on a set of isometric input data cubes versus using only object-centered cubes. To assess the value of multiple representations of sonar imagery resulting from isometries – *i.e.,* distance-preserving transformations – of each original input data example, we consider a set of 36 affine transformations that do not violate the physics of the sonar-object geometry. This set is formed from the Cartesian product of cross-track translations $i_{tx} = \{-0.1 \text{ m}, 0 \text{ m}, 0.1 \text{ m}\}$, along-track translations $i_{ty} = \{-0.1 \text{ m}, 0 \text{ m}, 0.1 \text{ m}\}$, cross-track reflections $i_{rx} = \{\pm 1\}$, and along-track reflections $i_{ry} = \{\pm 1\}$. The "centered input" case, in which the detected object is well-centered in the data cube, corresponds to $[i_{tx}, i_{ty}, i_{rx}, i_{ry}] = [0 \text{ m}, 0 \text{ m}, 1, 1]$.

Classification performance in terms of AUC for the eight trained CNNs is shown for the test data set in Table 4. Specifically, the AUC with and without using isometric input cubes is shown when considering all objects, only proud objects, or only buried objects. Also shown in the table is the ensemble performance, denoted $\mathcal{E}$, which uses, for a given test cube, the mean prediction of the eight CNNs as the final prediction. From the table, it can be seen that the use of isometric inputs consistently improves performance for each *individual* CNN, though not necessarily for the *ensemble* of CNNs. This result is an indication that the diversity engendered by the unique CNN architectures exceeds that which is created by the isometric inputs. Thus, in time-critical applications, one can accelerate the inference phase by obtaining classifier predictions

Table 4: AUC on the test set depending on whether isometric input cubes are used

	Only Centered Input Cubes Used			Isometric Input Cubes Used		
CNN	All Objects	Proud Only	Buried Only	All Objects	Proud Only	Buried Only
A	0.877	0.940	0.820	0.881	0.941	0.827
B	0.865	0.949	0.786	0.874	0.953	0.798
C	0.883	0.943	0.837	0.882	0.943	0.834
D	0.858	0.914	0.821	0.868	0.921	0.832
E	0.883	0.905	0.861	0.895	0.920	0.869
F	0.848	0.873	0.834	0.872	0.913	0.853
G	0.878	0.932	0.838	0.891	0.948	0.847
H	0.855	0.897	0.827	0.873	0.934	0.837
$\mathcal{E}(A\text{-}H)$	0.912	0.965	0.868	0.912	0.965	0.868

using only the object-centered cube, rather than the full set of isometries. Nevertheless, hereafter, all results correspond to the case using the set of 36 isometries.

Next, performance is presented in the form of full ROC-*like* curves, with the abscissa corresponding to the more informative false alarm *rate* instead of the *probability* of false alarm. The probability of false alarm is the probability of incorrectly classifying clutter as a target; the false alarm rate is the number of such incorrect classifications per image area or volume. When considering only proud objects, the false alarm rate is given per image (seafloor) area; when considering all objects or only buried objects, the false alarm rate is given per image volume.

Performance of the eight CNNs, as well as the ensemble, is shown in terms of ROC-like curves in Fig. 3. The AUC (for the corresponding ROC curve) is also provided in the legend. To provide a baseline measure of performance, the performance of a 3-d detection algorithm[6] – which makes predictions based on the geometric mean of a size feature and an intensity feature – is also shown. While the full curves are informative, in practice, one must select a single operating point at which to make predictions. Therefore, on each CNN curve, the operating point corresponding to a decision threshold of $\tau = 0.5$ is also marked.

As can be observed from Fig. 3, the 3-d CNNs greatly outperform the simple baseline detector, as would be expected. But more interestingly, the use of the ensemble of networks proves beneficial and removes the necessity of selecting a single best CNN architecture to employ. The complementary nature of the CNNs, and the unique clues that each uncovers and exploits to make predictions, leads to reduced false alarm rates. As a result, the ensemble approach can directly translate into cost savings during UXO remediation efforts.

5. CONCLUSION

A novel approach to the design of 3-d CNNs for underwater target classification tasks with volumetric SAS imagery was proposed. The careful design of the CNN architectures dramatically limits, by orders of magnitude, the number of free parameters that must be learned. This in turn eases the training data requirements, avoids problematic computer-memory issues, and greatly accelerates the CNN training phase. The promise of the approach was demonstrated on real, measured volumetric SAS data collected by an experimental system. This automated classification framework is well-suited for classifying both proud and buried man-made objects, such as UXO, in aquatic environments. Consequently, composing object "dig lists" based on the classifier predictions can lead to the efficient allocation of resources during remediation operations.

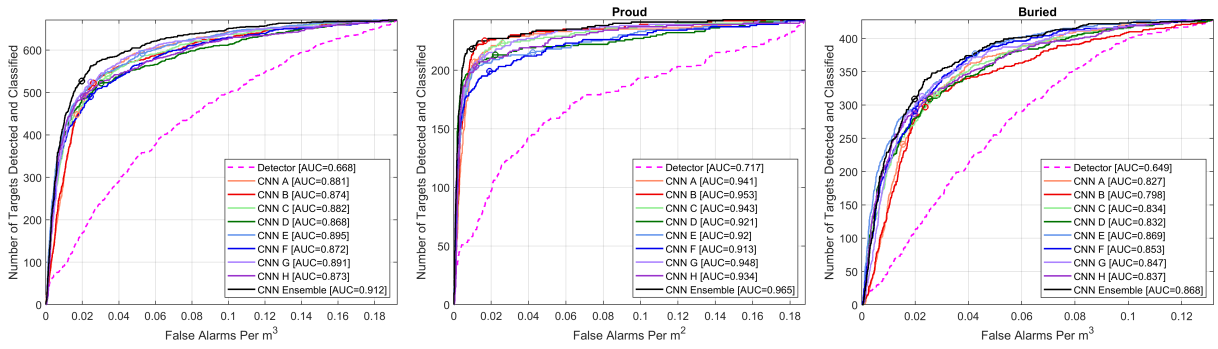


Figure 3: Performance on the SVSS test data set in terms of ROC-like curves for (a) all objects, (b) only proud objects, and (c) only buried objects. The operating point for a $\tau = 0.5$ threshold is marked with a circle.

ACKNOWLEDGMENTS

This research was supported by the U.S. Department of Defense, through the Strategic Environmental Research and Development Program (SERDP), under Projects MR18-1444 (DPW) and MR-2545 (DCB) in the Munitions Response area. With respect to the latter project, this material is based upon work supported by the Humphreys Engineer Center Support Activity under Contract No. W912HQ-16-C-0006.

REFERENCES

[1] S. Schock, A. Tellier, J. Wulf, J. Sara, and M. Ericksen, "Buried object scanning sonar," *IEEE Journal of Oceanic Engineering*, vol. 26, no. 4, pp. 677–689, 2001.

[2] D. Brown, S. Johnson, I. Gerg, and C. Brownstead, "Simulation and testing results for a sub-bottom imaging sonar," in *Proceedings of Meetings on Acoustics*, vol. 36, no. 1. Acoustical Society of America, 2019, pp. 1–12.

[3] T. Marston, D. Plotnick, and K. Williams, "Three dimensional fast factorized back projection for sub-sediment imaging sonars," in *Proceedings of IEEE OCEANS*, 2019, pp. 1–6.

[4] D. Brown, D. Cook, and J. Fernandez, "Results from a small synthetic aperture sonar," in *Proceedings of IEEE OCEANS*, 2006, pp. 1–6.

[5] A. Hunter and R. van Vossen, "Sonar target enhancement by shrinkage of incoherent wavelet coefficients," *The Journal of the Acoustical Society of America*, vol. 135, no. 1, pp. 262–268, 2014.

[6] D. Williams and D. Brown, "New target detection algorithms for volumetric synthetic aperture sonar data," in *Proceedings of Meetings on Acoustics*, vol. 40, 2020.

[7] I. Gerg, "The advanced synthetic aperture sonar imaging engine (ASASIN), a time-domain backprojection beamformer using graphics processing units," *The Journal of the Acoustical Society of America*, vol. 140, no. 4, pp. 3347–3347, 2016.

[8] J. Wallis and T. Miller, "Three-dimensional display in nuclear medicine and radiology," *Journal of Nuclear Medicine*, vol. 32, no. 3, pp. 534–546, 1991.

[9] D. Williams, "On the use of tiny convolutional neural networks for human-expert-level classification performance in sonar imagery," *IEEE Journal of Oceanic Engineering*, vol. 46, no. 1, pp. 236–260, 2021.

[10] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[11] J. Hanley and B. McNeil, "The meaning and use of the area under a receiver operating characteristic (ROC) curve," *Radiology*, vol. 143, pp. 29–36, 1982.

APPENDIX A

Here we present the intermediate responses at each layer of CNN D for one interesting data cube containing an 81 mm mortar that was buried 10 cm below the water-sediment interface. A photograph of this object (pre-burial) is shown in Fig. 4.

Figure 4: Photograph of an 81 mm mortar prior to burial at Sayers site B.

The intermediate responses at each layer of CNN D (trained on SVSS data) for an image cube containing this object are shown in Fig. 5. Because of the difficulty associated with displaying 3-d data, the intermediate responses are shown as a trio of MIPs in a common reference frame. However, it should be remembered that the input data and the intermediate responses are actually fully-populated 3-d volumes.

Examining intermediate responses like these can aid in the interpretation of what clues the filters are exploiting. For example, CNN D seems to be performing background removal, or equivalently, object segmentation.

APPENDIX B

To understand the sorts of filters these tiny 3-d CNNs learn, the convolutional filters (without the bias terms) of CNN D when trained on the SVSS data are shown in Fig. 6. Within each sub-figure, a common colorscale is used, in which the color green corresponds to zero, positive values are represented by warmer colors (*i.e.,* reds) and negative values are represented by colder colors (*i.e.,* blues). Each row corresponds to one filter. The 3-d filter cubes are separated in the z dimension (*i.e.,* depth into the sediment) solely to aid in visualization. For the 4-d tensors, filter depth runs horizontally across the page.

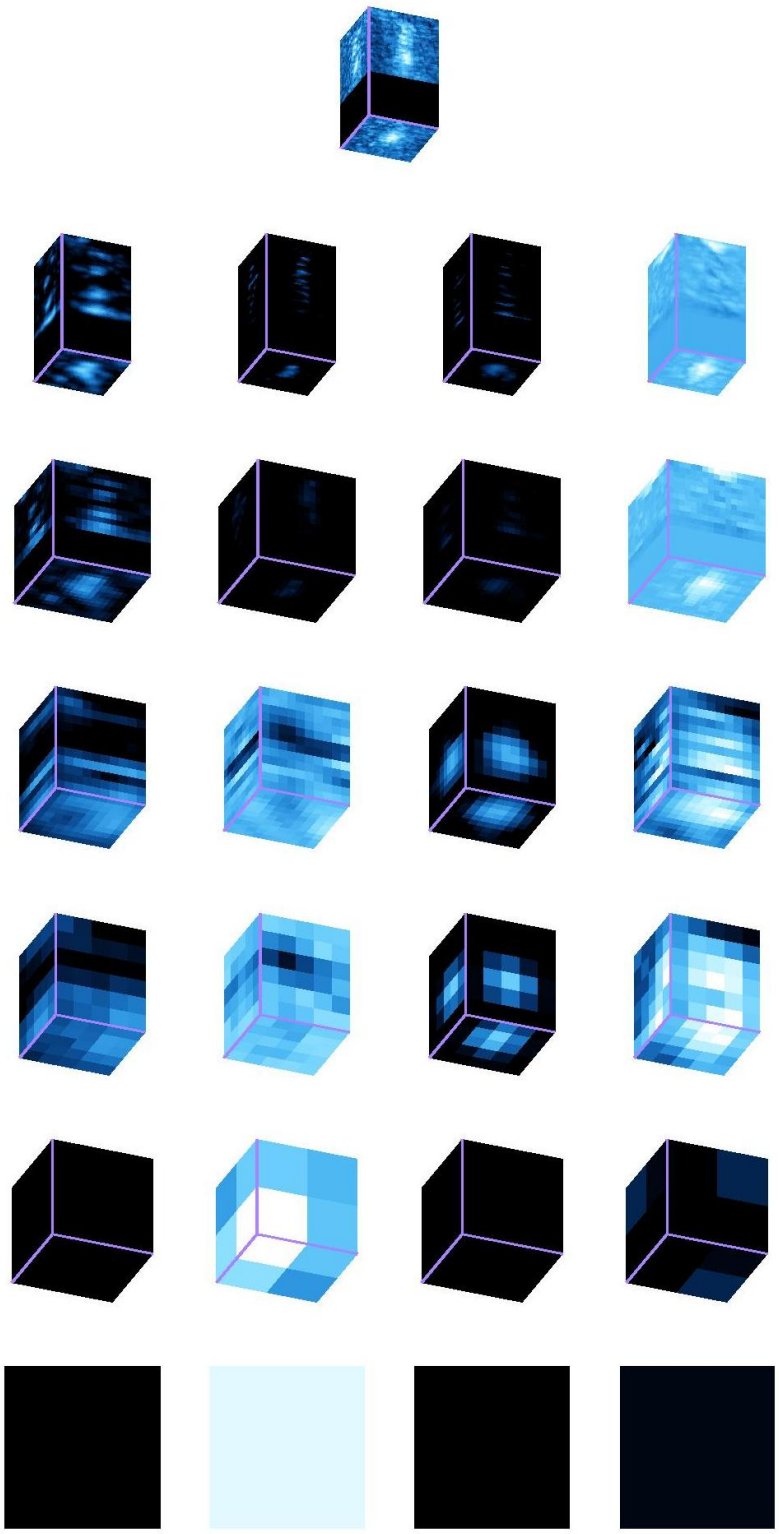
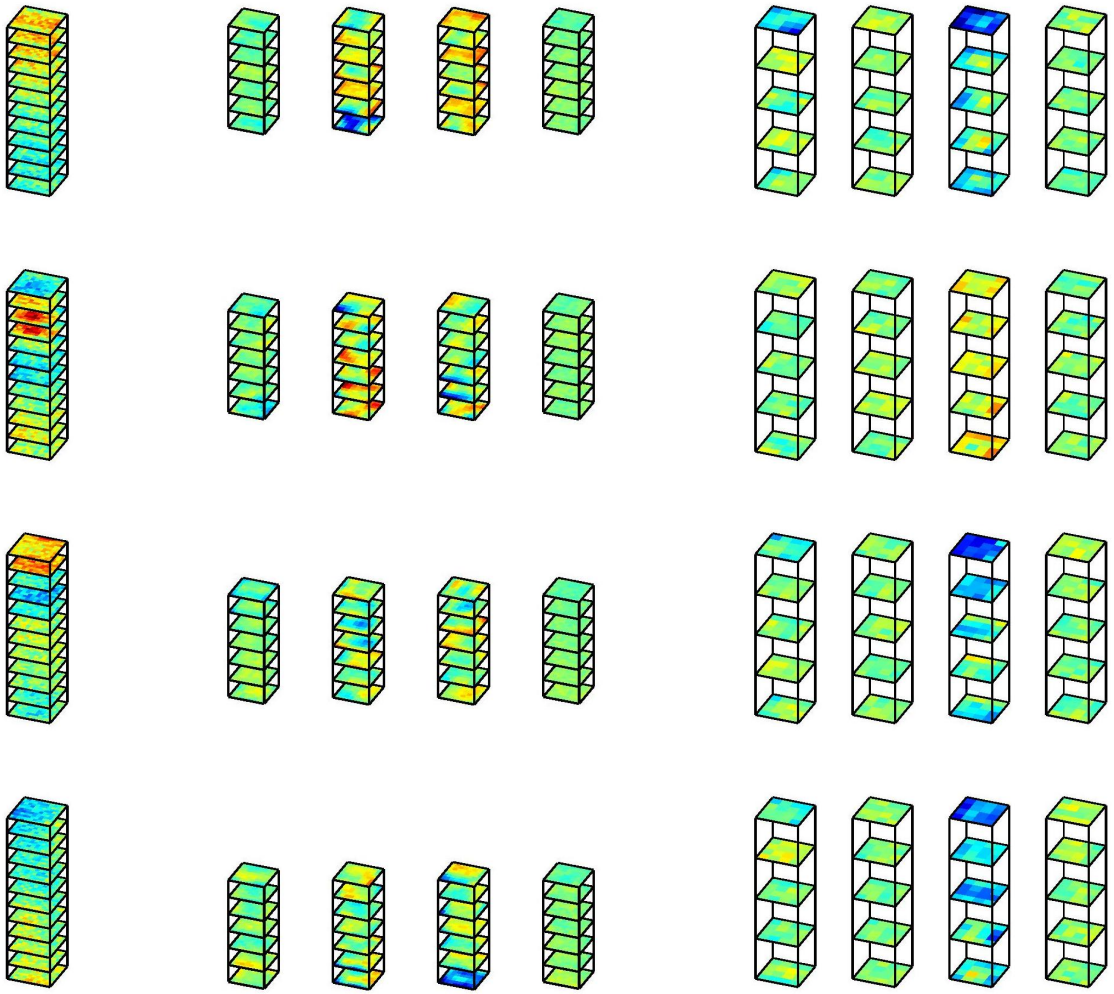


Figure 5: For the data cube in the top row, the intermediate responses (displayed as a trio of MIPs) at each layer of CNN D. Each row corresponds to the output at one layer (convolutional or pooling) of the CNN. The bottom row contains the set of 4 scalar features in the dense layer. With this CNN, the final probability of belonging to the target class was 0.98.



(a) Convolutional layer 1 (b) Convolutional layer 2 (c) Convolutional layer 3

Figure 6: The convolutional filters learned for CNN D using SVSS training data.