On the Use of Tiny Convolutional Neural Networks for Human-Expert-Level Classification Performance in Sonar Imagery

David P. Williams
NATO Science and Technology Organization
Centre for Maritime Research and Experimentation (CMRE)
Viale San Bartolomeo 400, 19126 La Spezia (SP), Italy
Phone: +39 0187-527-439, Fax: +39 0187-527-330
E-mail: david.williams@cmre.nato.int

Abstract—Efficient convolutional neural networks (CNNs) are designed and trained for an underwater target classification task with synthetic aperture sonar (SAS) imagery collected at sea. The main contribution is demonstrating that classification performance that matches, and even surpasses, the level achievable by a human domain-expert can be obtained from tiny CNNs with 3-6 orders of magnitude fewer parameters than have traditionally been used in the literature. In so doing, this work represents the first large-scale classification study in the sonar domain to establish a favorable comparison between automated algorithm performance and human ability. Extensive experimental results on challenging real-world SAS image data sets collected in diverse environments and conditions demonstrate that the CNNs possess strong generalization ability. These findings should significantly impact the manner in which CNNs are utilized in the underwater remote-sensing community. To wit, the tiny CNNs proposed here provide a blueprint for achieving excellent classification performance even with limited computing power or limited data.

I. INTRODUCTION

Convolutional neural networks (CNNs) [1] have recently achieved state-of-the-art performance on a wide range of image classification tasks [2]–[4] and they are quickly becoming the preferred image-classification method for any domain characterized by vast amounts of labeled data. And even in remote-sensing applications, where the sensor imagery available is typically limited, CNNs are also increasingly being employed [5]–[14].

The ascendance and supremacy of CNNs for image classification tasks has largely been due to the increased availability of data and processing power. Given sufficient amounts of these two key resources, a CNN will almost certainly outperform traditional “shallow” machine learning (ML) approaches. This observation is driven by the fact that traditional shallow classifiers eventually hit a performance plateau: beyond a certain point, incorporating more data ceases to improve performance [15]. In contrast, a CNN can continue to improve as more data is made available during the training phase [16], [17]. This relationship is shown in the cartoon in Fig. 1.

The scaling of CNN performance with training data is due to the richness of the CNN architecture, and specifically its enormous capacity, which permits more sophisticated decision surfaces than shallow classifiers [18]. Loosely speaking, the capacity of a CNN is dictated largely by the number of free trainable parameters in the model. The most popular architectures of famous “named” CNN families, such as VGG-Net [19], Inception [20], or ResNet [21], typically have between \(10^6\) and \(10^8\) free parameters. For comparison, commonly used shallow classification approaches, such as a support vector machine (SVM) [22] or relevance vector machine (RVM) [23], rarely have more than \(10^2\) or \(10^3\) free parameters, and can even have on the order of only 10 parameters.

The standard refrain about CNNs is that they require enormous amounts of data. But this lament elides subtle, but key, technical issues. The crucial quantity in determining the success of CNNs for classification tasks is not the amount of training data per se, but rather the relationship between training data and network capacity. As the number of free trainable parameters in the CNN model grows, so too do the training data requirements. (The complexity of the imagery itself, especially in the context of the number and types of classes to be discriminated, is also a factor.) If there is an insufficient amount of training data to support the model’s complexity (i.e., too many parameters), the model will overfit the training data, and performance on the test data will suffer. Therefore, when faced with limited training data, it is imperative to constrain the CNN’s capacity by employing smaller networks. This notion, illustrated in the cartoon in
Fig. 2, is at the crux of statistical learning theory [18], [24, Chapter 5]. (However, it should be noted that explaining the generalization ability of deep-learning techniques is still a very active area of research, and one in which more modern perspectives are contributing [25]–[30].)

In this work, we leverage these insights to design and train tiny CNNs containing orders of magnitude fewer parameters than have been traditionally used in the literature. We demonstrate that classification performance using these CNNs can match, and even surpass, the level achievable by a human domain-expert on real underwater target classification tasks with synthetic aperture sonar (SAS) imagery.

Much of the previous work using CNNs for SAS-image classification tasks has instead focused on taking pre-trained CNNs designed for – and trained with – optical imagery and transferring them to the new (sonar) domain. Some early work considered the use of pre-trained CNNs as feature extractors. In [31], pre-trained AlexNet and VGG-Net CNNs were used to obtain features for an SVM classifier in order to distinguish four classes of object shapes. A similar investigation for the binary target/clutter classification problem was undertaken in [32] using only the AlexNet CNN. Subsequent studies began to compare the performance of the pre-trained CNNs to new CNNs trained from scratch on SAS data, but findings have been inconsistent. For example, in [33], various approaches for fine-tuning AlexNet and VGG-Net CNNs were investigated, and it was found that refining all layers of the networks (and thus learning millions of parameters) was necessary to achieve the best performance on a target classification task. Additional experiments in that work showed that refining pre-trained VGG-16, DenseNet-161, Inception ResNet, and NasNet CNNs achieved better performance than using two custom CNNs trained from scratch; the custom CNNs had only 4 or 5 convolutional layers, yet there were still approximately $3 \times 10^5$ parameters to learn in each. But in [34], training new custom CNNs from scratch was compared to fine-tuning a pre-trained VGG-16 CNN in the context of a seafloor classification task. In that work, the custom CNNs, composed of 8 convolutional layers and around $3 \times 10^6$ total parameters, outperformed the pre-trained CNNs. The performance when using different training techniques (e.g., data augmentation, optimization) and pooling approaches was also examined, but the effects of network depth and filter sizes were not. In [35], training new ResNet CNNs (with 18 to 50 layers involving convolutions, and around $10^7$ parameters for the former) from scratch with dual-band SAS data was found to perform essentially the same as fine-tuning the last layer of ResNet-50 CNNs that had been pre-trained with optical imagery. In [36], a custom CNN with only 2 convolutional layers, but over $2 \times 10^5$ parameters, was considered for a target classification task.

To summarize, the custom CNNs developed in [33]–[36] for a task and data modality similar to what is considered in this work employed usually many tens or hundreds of filters per convolutional layer, (sometimes multiple) fully-connected layers with many nodes (up to 1024), no more than 8 convolutional layers (for the non-ResNet cases), and on the orders of $10^2$ to $10^3$ parameters to learn. In contrast, the novel CNNs we design in this work contain only 4 filters per convolutional layer and only a single fully-connected layer (of only 4 nodes), but up to 20 convolutional layers and on the orders of only $10^2$ to $10^3$ parameters to learn.

Many previous approaches concerned about model size begin with a typical, million-plus-parameter CNN but then institute a pruning procedure to progressively remove filters, nodes, or connections from the network based on some criterion [37]–[40]. Another common way to constrain network size is through the use of shallow, but wide CNNs – often trained layer-wise – that feature relatively few convolutional layers but sufficiently many filters [41]–[43]. But to our knowledge, no one has exploited modest-depth CNNs while simultaneously reducing the number of filters to the drastic extent that we propose in this work.

Although the work in this paper extends and leverages the findings of our previous deep-learning research in [44]–[46], the material presented here is new. The new set of CNNs employed here is specially designed to enable a substantive study of the effects of CNN depth, filter size, and pooling factors on performance. The comparison to human-expert performance and the related analysis is also novel. To our knowledge, no one has exploited modest-depth CNNs while simultaneously reducing the number of filters to the drastic extent that we propose in this work.

The remainder of this paper is organized as follows. Sec. II explains CNNs, but with a concerted emphasis placed on making the concept more accessible to the wider research community; the architectures of the newly designed CNNs are also detailed. Sec. III describes the CNN training procedure undertaken, and the measured SAS data used to perform it. Sec. IV presents experimental results on the underwater-target classification task. Concluding remarks and directions for future research are given in Sec. V. Supplemental material regarding the learned CNNs and additional results of the human labeling experiments is provided in three appendices.
II. CONVOLUTIONAL NEURAL NETWORKS

A. A Gentle Overview

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., targets and clutter). Within these bookends, the architecture of a CNN consists of a sequence of layers, each 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.

A standard CNN typically consists of convolutional layers, nonlinear activation functions, and pooling layers, that are then followed by one or more fully-connected “dense” layers, and a final prediction. A schematic representation of this basic architecture is shown in Figure 3.

Training a CNN simply means learning the filters, and associated bias (i.e., offset) terms, of the convolutional layers.1 (There are no parameters associated with the pooling layers.) Instead of using pre-defined filters manually created by a human, the CNN automatically learns what the filters should be from the data, and in effect, the most useful bases in which to represent the data. That is, CNNs obviate the extraction of predefined, human-engineered features, which can be challenging for difficult-to-interpret data representations (e.g., acoustic color). This property makes CNNs particularly appealing, as it allows the algorithm to uncover clues beyond those enumerated by human intuition. One perceived drawback that accompanies this power is the challenge of explaining CNN decisions, a topic receiving attention [47].

The filters transform the input data (imagery) into a new representation space. This new representation is simply the output of convolving the filters, also known as kernels, with the input image. The intermediate representations of the data in each CNN layer are usually referred to as feature maps. (When there are multiple feature maps in a layer, the associated filters will be three-dimensional tensors.)

With successive convolutional layers, the level of data abstraction and (highly nonlinear) decision-surface complexity increases. (The number of convolutional layers employed, known as the depth of the network, also plays a major role in determining the capacity of the CNN.) The data representation at the penultimate dense layer can be viewed and treated as features, in a manner analogous to the features that are extracted—from original image data, for use as inputs—for a shallow classifier. In essence, what a CNN does is learn a transformation to map an input image into a new representation space in which the classes are easily separable.

During the training process, the model seeks to minimize an objective (or “loss”) function expressing the classification error on the training data under consideration. At each training iteration, the model parameters are updated by some variant of gradient descent in order to get closer to this objective. Because there can be huge numbers of free model parameters to be learned, it is necessary to have a correspondingly large set of training data to avoid overfitting. In turn, training a CNN ab initio can take considerable time, even with high-throughput computational resources like graphics processing units (GPUs). Since the amount of SAS data is always increasing (e.g., with each data-collection survey), CNNs are a convenient way to leverage and exploit the fruits of these investments fully.

Designing a CNN architecture is often considered an art form, and domain expertise is vital to creating successful networks. Assuming a standard “vanilla” architecture, the key attributes that a designer must decide are the number of convolutional layers to employ, the number of filters to be learned in each convolutional layer, and the sizes of those filters. The designer must also decide where in the architecture to place pooling layers (which encourage translation invariance), as well as the pooling factors to use in those layers, and the manner in which the pooling is performed (e.g., taking an average or the maximum value in each region). The size of the input data, the manner in which the input data is normalized, the number of nodes in the dense layer, and the mathematical form of the activation functions must also be specified. The training process entails several additional choices, including the exact form of the objective function (which will not be convex), the optimization algorithm and various parameter settings therein (such as learning rate and batch size), and data augmentation strategies to employ, among other things. Classification performance can be very sensitive to many of these design choices. A more thorough, book-length discussion of CNNs can be found in [24].

B. Mathematical Details

Having given this qualitative overview of CNNs, we now define the most important quantities more precisely for the binary classification case with single-channel input images. The
following presents the equations that govern the operations in each layer of a CNN, parameter training, and how one obtains final classification predictions for an input image.

Let \( x \) be an input image of an object associated with class label \( y \in \{0, 1\} \), where \( y = 1 \) indicates a target and \( y = 0 \) indicates clutter.

Let \( a^{(\ell-1)}_i \) be the \( i \)th feature map in the \((\ell-1)\)th convolutional layer. Let \( w_{ij}^{(\ell)} \) denote the convolutional kernel connecting the \( i \)th input feature map to the \( j \)th output feature map in the \( \ell \)th convolutional layer, and let \( b_{ij}^{(\ell)} \) denote the bias in the \( j \)th feature map in the \( \ell \)th convolutional layer.

The response of the \( j \)th output feature map at the \( \ell \)th convolutional layer is obtained as

\[
a_j^{(\ell)} = \sigma(z_j^{(\ell)}), \quad z_j^{(\ell)} = \sum_i a_i^{(\ell-1)} * w_{ij}^{(\ell)} + b_{ij}^{(\ell)}
\]  

(1)

(2)

where \( * \) denotes the convolution operation, and \( \sigma(\cdot) \) is the nonlinear activation function. At the CNN input, there is only one feature map: \( a_j^{(0)} = x \).

Two nonlinear activation functions used in this work are the sigmoid function and the rectified linear unit (ReLU), which are given by

\[
\sigma(z(r,c)) = (1 + \exp(-z(r,c)))^{-1} \quad (3)
\]

\[
\sigma(z(r,c)) = \max\{0, z(r,c)\} \quad (4)
\]

respectively; \( (r,c) \) specifies a row and column position in a feature map.

Pooling layers reduce the spatial size of feature maps in accordance with pooling factors, \( f_R \) and \( f_C \), specified for each dimension. Two common types of pooling are average pooling and max pooling. In average pooling, a sub-region of a feature map is mapped to its average value,

\[
a_j^{(\ell)}(r,c) = \frac{1}{f_R f_C} \sum_{u=1}^{f_R} \sum_{v=1}^{f_C} a_j^{(\ell)}(r + u - \lceil f_R/2 \rceil, c + v - \lceil f_C/2 \rceil),
\]

(5)

where \( \lceil \cdot \rceil \) is the mathematical ceiling. In max pooling, a sub-region of a feature map is mapped to its maximum value,

\[
a_j^{(\ell)}(r,c) = \max_{1 \leq u \leq f_R, 1 \leq v \leq f_C} a_j^{(\ell)}(r + u - \lceil f_R/2 \rceil, c + v - \lceil f_C/2 \rceil).
\]

(6)

At the final output layer \( L \) of the CNN, the softmax function gives the probability of the input image \( x \) belonging to each class \( y_i \),

\[
p(y_i|x^{(L)}) = \text{exp}\left\{z_i^{(L)}\right\} \sum_{j=1}^{J} \text{exp}\left\{z_j^{(L)}\right\}.
\]

(7)

The binary-cross-entropy loss function for one training sample, \( \{x; y\} \), is defined as

\[
J(w, b) = -y \log p(y|x^{(L)}; w, b) - (1 - y) \log(1 - p(y|x^{(L)}; w, b)).
\]

(8)

When a “mini-batch” of \( B \) training samples is employed, the loss function simply becomes the mean loss over the \( B \) samples.

Mini-batch gradient descent updates the parameter values according to

\[
w \leftarrow w - \eta \frac{\partial J(w, b)}{\partial w},
\]

(9)

\[
b \leftarrow b - \eta \frac{\partial J(w, b)}{\partial b},
\]

(10)

where \( \eta \) is the learning rate that controls the size of the update step, and \( J(w, b) \) is the loss function of the mini-batch. At each iteration of the training procedure, a new mini-batch of training samples is randomly selected.

The RMSprop optimizer [48] is simply a “fancy” way to make the learning rate adaptive; it normalizes the learning rate for a parameter by a running average of the magnitudes of recent gradients for that parameter.

The backpropagation algorithm [49], which relies on the chain rule from calculus, is used to compute the necessary partial derivatives in a CNN in a computationally efficient manner.

C. CNN Design

In this work, we design eight CNNs that share a common architecture of alternating convolutional blocks (comprising one or more convolutional layers) and pooling layers; a high-level diagram is shown in Fig. 4. The input to a CNN is assumed to be a 267 pixel \( \times \) 267 pixel SAS magnitude image, where each pixel spans 1.5 cm in each dimension. All pooling layers use average pooling, rather than max pooling, because the former approach has been observed [34], [44] to better handle the speckle phenomenon that characterizes sonar imagery. The outputs of the CNN’s final layer are the (softmax) probabilities of an image belonging to each class (target or clutter).

The architectures are designed with an eye toward investigating classification performance as a function of CNN depth, and to observe the impact of filter size and pooling factors. The depths of the CNNs considered range from 4 to 20 convolutional layers. For each of the 4, 8, and 12 convolutional layer cases, two unique CNNs are designed; one uses smaller filters and larger pooling factors, while the other employs larger filters and smaller pooling factors.

Details about the specific CNNs are provided in Table I; 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 \( i \)th set of brackets contains the information about the convolutional layers in
the $i$th block. For example, the first convolutional block of CNN D comprises a convolutional layer with $8 \times 8$ filters, followed by a second convolutional layer with $6 \times 6$ filters.) The convolutional block construct allows deeper networks, and thus greater complexity, without a proportional increase in the number of parameters to learn.

Each CNN contains 4 convolutional blocks; each block contains a specific number of convolutional layers (equal to the number of rows in Table I’s filter-size column). Each filter 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. 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 in order to scrutinize the classification power of small networks when faced with limited training data. Collectively, the eight CNNs have only 26480 parameters to learn, which is still 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 of filters per convolutional layer. 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 $1 \times 1$ 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, which has been shown to be a key driver of performance [50].

The overarching philosophy in the ML literature is to design huge networks but then incorporate some regularization technique [24, Chapter 6] – such as drop-out [51], which randomly masks nodes in a network – as a compensatory mechanism to prevent overfitting. In contrast, we design our networks to be efficient from the start, which also reduces computing-resource requirements. We believe the binary nature of the classification task being addressed in the underwater domain – compared to the huge multi-class problem [52] addressed in the computer-vision community with optical images – and other characteristics of SAS data (e.g., lower resolution, smaller training sets) justify these CNN design choices.

III. SAS CNNs

A. SAS Data

SAS relies on the coherent processing of acoustic returns to produce high-resolution imagery of underwater environments that can be exploited for object classification and other tasks [53]. A relatively large database of scene-level SAS images, each of which typically spans 50 m in the along-track direction and 110 m in the range direction, had been collected by CMRE’s MUSCLE autonomous underwater vehicle (AUV) [54] during thirteen sea expeditions conducted between 2007 and 2017 in various geographical locations. The center frequency of the SAS is 300 kHz and the bandwidth is 60 kHz. The imagery has an along-track resolution of 2.5 cm and a range resolution of 1.5 cm. (The imagery gets upsampled to a 1.5 cm along-track resolution before it is used in the CNNs.) Prior to the surveys, mine-like object shapes including cylinders, truncated cones, wedges, and other man-made objects had been purposely deployed. This allowed detailed target ground-truth information to be formulated. The data of the different expeditions vary greatly in terms of seafloor composition and cover (e.g., sand, mud, vegetation), clutter types and densities, target types, image quality, and environmental complexity. Data from the eight oldest expeditions were treated as training data for learning CNNs, while the data from the five most recent expeditions were reserved for use as distinct test sets (owing to their diverse characteristics).

A summary of the MUSCLE data sets is given in Table II. In the table, the target types $T_1$ through $T_5$ correspond to cylinders, truncated cones, wedges, two specially calibrated rocks, and all remaining miscellaneous target objects, respectively. (The miscellaneous object set, $T_6$, contains diverse items including a washing machine, a scuba dive bottle, a tire, a weighted gym bag, and an AUV glider, among others.) However, because of the dearth of target data, only the binary classification problem – target or clutter – is considered. From the table, it can be observed that severe class imbalance exists, and that the training set is limited by the amount of target data available.

The test data sets are quite diverse in terms of the distribution of target types, but also in terms of environment. The MAN2 set contains considerable environmental clutter as well as regions of sand ripples, in which many of the targets lie. The NSM1 set suffers from poor image-quality due to strong currents encountered during data collection; large sand mounds are also present. The seafloor of the TJM1 set was flat sand, that of the ONM1 set comprised small shells, and that of the GAM1 set was of softer mud and silt. The differences among the test data sets can be observed more readily in example SAS images shown later.

The Mondrian detection algorithm [55] was applied to all scene-level sonar images in the database. This algorithm is a flexible, general-purpose object-detection method that can reliably detect objects over the wide range of potential target sizes and shapes. The detector results in a set of image “chips” of objects to be classified as targets (class 1) or clutter (class 0) by the CNN classifiers; each chip spans $5.025 \text{ m} \times 5.025 \text{ m}$, larger than the requisite CNN input size, to facilitate data augmentation (explained later).

From the detector, an object’s highlight region is typically well-centered within a chip. However, the accompanying shadow region (cast because of the object’s height above the seafloor) sometimes exceeds the bounds of a chip. This result arises from a trade-off to constrain the chip size (with which
CNN computation scales), while still retaining local contextual (background) information; but it is also a compromise between extreme short-range and long-range detections (since shadow length of a given object will scale with range from the sensor), and a way to limit the fraction of chips that exceed a scene-level image’s natural boundary (a mirroring operation is performed in the Mondrian detector to artificially expand scene-level imagery, with this allowing full chips of “partial” targets near an edge to still be detected and extracted).

It is germane to note that a pre-processing step in the detection algorithm normalizes the scene-level imagery, from which the chips are extracted; this entails median-normalization, a logarithmic transformation, and a clipping operation (see Sec. II.F. in [55] for details). Before CNN training or evaluation, a second normalization scheme is then applied to the chips to transform the pixel values from $[0, 40]$ to $[-1, 1]$, according to the simple mapping

$$x'_{ij} = x_{ij}/20 - 1,$$

where $x_{ij}$ is the $ij$th pixel in the chip $x$.

### B. CNN Training

CNN training was performed in Python with the TensorFlow [56] 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 inviolable geometry of the sonar data-collection procedure was employed during training; this meant a random range translation \(i_{tx} \in [0 \text{ m}, 0.5 \text{ m}]\), along-track translation \(i_{ty} \in [-0.5 \text{ m}, 0.5 \text{ m}]\), and along-track reflection \(i_{ry} \in \{0, 1\}\) was applied, “on-the-fly,” to each SAS image chip selected for the batch. This range-translation interval was chosen in order to avoid reducing the chip’s shadow region.

Each CNN was trained for 100 epochs, where one epoch was defined as a set of 1000 batches. Each batch was formed by randomly selecting chips from the full set of training data. (Thus, during the CNN training phase, each of the 2912 training-set target chips is seen about 1100 times and each of the 652514 training-set clutter chips is seen about 5 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. The progression of the training loss (computed only from the data used to select the final model to evaluate the test sets. The model achieved the maximum AUC on the validation set was chosen in order to avoid reducing the chip’s shadow region. Thus, during the CNN training phase, each of the 2912 training-set target chips is seen about 1100 times and each of the 652514 training-set clutter chips is seen about 5 times, on average.)

Each CNN was trained for 100 epochs, where one epoch was defined as a set of 1000 batches. Each batch was formed by randomly selecting chips from the full set of training data. (Thus, during the CNN training phase, each of the 2912 training-set target chips is seen about 1100 times and each of the 652514 training-set clutter chips is seen about 5 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. The progression of the training loss (computed only from the data used in a given epoch’s batches) and the validation AUC as a function of epoch for each of the eight CNNs is shown in Fig. 5. (For reference, training time – when the data is stored and read locally – for these CNNs is about 1 second per batch with a single GeForce GTX 1080 GPU; speed optimization is not the focus of this work, but significant acceleration is certainly possible.)

For a general idea of what sorts of filters these tiny CNNs learn from the sonar data, the convolutional filters of CNN E are shown in Appendix A. It can be noted that the filters do not resemble Gabor functions the way many CNNs trained on optical images do [2].

IV. 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) [57], 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 curves would need to be scaled by the proportion of targets actually found in the detection stage. Finally, a note about nomenclature: On all figures showing ROC (and ROC-like) curves, the term “probability of detection” is used in the classical signal-detection theory sense [58] of discriminating a target from clutter; thus, the quantity being measured is in fact the performance of the classification stage.

To obtain a baseline measure of performance, we use the Mondrian detection algorithm, which makes predictions based on a small set of traditional highlight and shadow features (with fixed weights), as a “shallow” classifier. (In our experience, the Mondrian algorithm is more robust and performs better on this sensor’s data than conventional shallow classifiers like an SVM or RVM, which generalize poorly and are highly-dependent on the training data used.)

B. Human Domain-Expert

Perfect classification performance on the MUSCLE test data sets is unrealistic because of errors in the ground-truth, unrecognizable target signatures due to environmental occlusions (e.g., targets physically covered by posidonia or concealed in ripple shadow zones), poor image quality, and other miscellaneous factors. Therefore, to estimate the best possible performance actually achievable, we enlist 12 human subjects with extensive experience working with SAS images for MCM. Drawing inspiration from the study described in [52], in which two human annotators were tasked with labeling optical images in the ImageNet database, we undertake a similar experiment. (In the study in [52], the two human annotators labeled only 1500 and 258 test images, respectively, though for a much larger multi-class problem.)

We begin with the ONM1 test set due to its modest size. Each of the 186 SAS images in this test set is randomly presented one at a time to a human subject, who then hand-labels each as target or clutter. The outcome of this process provides a single operating point on a hypothetical ROC curve (namely, at a decision threshold of \(\tau = 0.5\)). Although it is unrealistic to rank-order all of the images in a given test set, an alternative way to obtain a full ROC curve for a human subject is possible, as follows.

The labeling operation for a given image chip is a two-alternative forced-choice (TAFC) task, and a fundamental assumption in such tasks is that a “decision is made when sufficient evidence has accumulated favoring one alternative over the other” [59]. In our context, this means that the longer it takes the human subject to make a decision, the more difficult it had been to uncover in the image the necessary clues to make a decision, and hence the more challenging the image was to classify. That is, one could argue (and assume) that there is a direct relationship between classification confidence and the speed with which the classification decision is made. Therefore, in addition to recording a human subject’s class prediction – \(y_i = 1\) (target) or \(y_i = 0\) (clutter) – for the \(i\)th image chip, the time taken to make the decision, \(t_i\), is also recorded. Each binary decision of the human is then transformed into a continuous-valued confidence score in [0, 1] that the image belongs to the target class according to

\[
p_i = \frac{1}{2} \left( 1 + (-1)^{y_i+1} e^{-t_i} \right).
\]

This mapping ensures that all images labeled as targets will have a target confidence greater than 0.5, while all images labeled as clutter will have a target confidence less than 0.5. Additionally, it produces a reasonable rank-ordering of the images, so that a full ROC curve for a human subject can be constructed. (The humans were informed in advance of the decision time’s importance and performed the labeling of each test set in a single sitting.)

Then, based on the results from this smallest test set, the subject with the best performance (and who was also available
C. Classification Results

Classification performance of the eight trained CNNs is assessed in several ways. First, we examine the benefit of making predictions based on a set of isometric input images versus using only image-centered inputs with a CNN.

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 18 affine transformations that do not violate the physics of the sonar-object geometry. This set is formed from the Cartesian product of range translations \( i_{tx} = \{ 0 \text{ m}, 0.25 \text{ m}, 0.50 \text{ m} \} \), along-track translations \( i_{ty} = \{-0.25 \text{ m}, 0 \text{ m}, 0.25 \text{ m} \} \), and along-track reflections \( i_{ty} = \{ \pm 1 \} \). The “centered input” case (i.e., 1-isometry case), in which the detected object is well-centered in the imagery, corresponds to \([i_{tx}, i_{ty}, i_{ty}] = [0 \text{ m}, 0 \text{ m}, 1]\).

We also consider intermediate cases in which only 2, 4, or 8 isometries are used. The intuition dictating the specific isometries considered was to select the subset that collectively would exhibit the most diversity, while also satisfying the constraint that the full object signature was retained in each image. This approach was expected to yield the best performance most efficiently (i.e., with the fewest isometries) so that computation time during evaluation could be minimized.

The 2-isometry case forms the set from the Cartesian product of range translations \( i_{tx} = \{ 0 \text{ m} \} \), along-track translations \( i_{ty} = \{ 0 \text{ m} \} \), and along-track reflections \( i_{ty} = \{ \pm 1 \} \).

The 4-isometry case forms the set from the Cartesian product of range translations \( i_{tx} = \{ 0 \text{ m}, 0.25 \text{ m}, 0.50 \text{ m} \} \), along-track translations \( i_{ty} = \{ 0 \text{ m} \} \), and along-track reflections \( i_{ty} = \{ \pm 1 \} \).

The 8-isometry case adds to the 4-isometry case the set from the Cartesian product of range translations \( i_{tx} = \{ 0 \text{ m}, 0.125 \text{ m}, 0.25 \text{ m}, 0.375 \text{ m}, 0.50 \text{ m} \} \), along-track translations \( i_{ty} = \{ 0 \text{ m}, \pm 0.125 \text{ m}, \pm 0.25 \text{ m} \} \), and along-track reflections \( i_{ty} = \{ \pm 1 \} \). Finally, an extreme 50-isometry case was also considered, using the set from the Cartesian product of range translations \( i_{tx} = \{ 0 \text{ m}, 0.125 \text{ m}, 0.25 \text{ m}, 0.375 \text{ m}, 0.50 \text{ m} \} \), along-track translations \( i_{ty} = \{ 0 \text{ m}, \pm 0.125 \text{ m}, \pm 0.25 \text{ m} \} \), and along-track reflections \( i_{ty} = \{ \pm 1 \} \). For a given test chip, the mean prediction for the set of isometric images considered is used as the final prediction. (Each isometric image in the set is submitted to the CNN for evaluation one at a time.)

Classification performance in terms of AUC for the eight trained CNNs is shown for the five test data sets in Tables III-VII. Specifically, the AUC as a function of the number of isometric input images used is shown. Also shown in each table is the ensemble performance, denoted \( E \), which uses, for a given test image, the mean prediction of the eight CNNs as the final prediction. From the tables, it can be seen that the use of multiple isometric inputs, up to 18, consistently improves performance for all CNNs across all test sets. But in particular, averaging over a set of predictions has the largest impact (i.e., performance improvement) on the most challenging data set, NSM1. The difference in performance between the 18-isometry and 50-isometry cases was negligible, and does not justify the attendant increase in computational cost from the additional evaluations. Therefore, hereafter, all results correspond to the case using the set of only 18 isometries.

These results show the benefit of extracting, from the SAS image scene, a larger chip than that required as input to the CNNs. The larger chip enables the easy formation of the isometries, which then result in a more robust final CNN prediction. Additionally, it can be seen that using an ensemble of CNNs with unique architectures induces a larger performance gain than using multiple isometric inputs for a single
CNMs, but that the best performance is achieved when both strategies are employed. Nevertheless, when relying on the ensemble of CNNs, the number of isometries can be reduced to 4 from 18 with barely any degradation in performance. That is, worthwhile performance gains can be realized inexpensively by considering only a modest number of isometries.

Next, we more closely examine the impact of CNN design – namely, the number of convolutional layers, the size of the filters therein, and pooling factor sizes – on classification performance. To do so, performance is presented in the form of an ROC-like curve, 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 wide-area scene-level image.) However, in data sets characterized by severe class imbalance, as these do, another commonly used metric is the precision-recall (PR) curve, which places more focus on the minority class (here, targets). Precision is defined as the fraction of images classified as targets that are actually targets, while recall is defined as the fraction of targets that are correctly classified as such (and thus is identical to the probability of detection). The area under the precision-recall curve, which we denote AU(PR), can also be computed as a summary measure of performance much like the AUC provides a summary for ROC curves; higher AU(PR) values are better, and the maximum is unity. In the context of precision-recall curves, another useful quantity is the $F_1$ score, which is the harmonic mean of the precision and recall, and thus is a metric that balances the trade-off between the two components.

Performance of the eight CNNs, as well as the ensemble, is shown in terms of full ROC-like curves and precision-recall curves in Figs. 6-10. The AUC (for the corresponding ROC curve) and AU(PR) are also provided in the legends. In addition to the CNNs, the performances of the baseline Mondrian detection algorithm and of the human expert are 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 curve, the operating point corresponding to a decision threshold of $\tau = 0.5$ is also marked.

From the figures, it can be seen that the tiny CNNs perform
relatively well individually, but that collectively – as an ensemble – they typically match or surpass the performance of the human domain-expert. The one exception is the NSM1 data set, where the human performance is still superior. This data set is extremely challenging as it is largely characterized by severely poor image quality. This fact was due to the presence of strong currents in the North Sea during the sea expedition, coupled with the fact that the data-collection surveys were designed to test AUV autonomy algorithms, rather than to collect optimal data for strict classification missions. None of the sea expeditions in the training data set shared such extreme conditions, and these results underscore the importance of sufficient diversity in one’s training data. (The CNNs do still greatly outperform the shallow classifier on the NSM1 data set.) Nevertheless, in the four other test sets, the ensemble of CNNs achieves par-human-expert performance, despite having a collective total of only 26480 parameters. That is, by relying on orders of magnitude fewer parameters to learn than in the most popular CNN architectures, the set of tiny CNNs can attain impressive performance.

To more easily see the overall performance of the different CNNs (and competing methods) on the five test sets, the AUC and AU(PR) are collected and presented visually in Fig. 11 as a function of the number of CNN parameters. From this figure, the power of the CNN ensemble is more readily apparent. One can also observe that in general, the CNNs that employ smaller filters but larger pooling factors – CNNs A, C, and E – tend to perform better than their respective counterparts with the same depths (but larger filters and smaller pooling factors). (The mean AUC of CNNs A, C, and E is higher than the mean AUC of CNNs B, D, and F on four of the five test sets.) Although the ensemble of CNNs performs best, the single best CNN overall appears to be CNN E, which had a depth of 12 convolutional layers (and only 3289 parameters).

Next, a summary of the key performance metrics of different methods is provided in Table VIII. (The quantity “FA / Image” stands for false alarms per scene-level image, from which a false-alarm density – i.e., per unit area – can be computed easily.) Specifically, two metrics that summarize performance over the full range of operating points are shown, and four metrics computed at a single key operating point – corresponding to a decision threshold of $\tau = 0.5$ – are given. From the table, the superiority of the ensemble of CNNs over the shallow Mondrian algorithm (on all five test sets) and over the human expert (on four of the five test sets), in terms of AUC and AU(PR), is readily apparent. The false alarm rates at the $\tau = 0.5$ threshold succinctly reveal the relative difficulty of the different test sets.
Fig. 6. Performance on the NSM1 test data set, in terms of (a) ROC-like curve and (b) precision-recall curve. The operating point for a $\tau = 0.5$ threshold is marked with a solid dot. In (a), the AUC for the corresponding ROC curve (i.e., in which the abscissa represents probability of false alarm instead of false alarm rate) is given in the legend. In (b), $A$ in the legend indicates the area under the precision-recall curve.

Fig. 7. Performance on the MAN2 test data set, in terms of (a) ROC-like curve and (b) precision-recall curve. The operating point for a $\tau = 0.5$ threshold is marked with a solid dot. In (a), the AUC for the corresponding ROC curve is given in the legend. In (b), $A$ in the legend indicates the area under the precision-recall curve.
Fig. 8. Performance on the TJM1 test data set, in terms of (a) ROC-like curve and (b) precision-recall curve. The operating point for a $\tau = 0.5$ threshold is marked with a solid dot. In (a), the AUC for the corresponding ROC curve is given in the legend. In (b), $A$ in the legend indicates the area under the precision-recall curve.

Fig. 9. Performance on the ONM1 test data set, in terms of (a) ROC-like curve and (b) precision-recall curve. The operating point for a $\tau = 0.5$ threshold is marked with a solid dot. In (a), the AUC for the corresponding ROC curve is given in the legend. In (b), $A$ in the legend indicates the area under the precision-recall curve.
Fig. 10. Performance on the GAM1 test data set, in terms of (a) ROC-like curve and (b) precision-recall curve. The operating point for a $\tau = 0.5$ threshold is marked with a solid dot. In (a), the AUC for the corresponding ROC curve is given in the legend. In (b), $\mathcal{A}$ in the legend indicates the area under the precision-recall curve.
Fig. 11. For the MUSCLE test data sets, performance summary for each CNN in terms of (a) AUC and (b) AU(PR). The black dash-dotted line indicates the CNN ensemble performance, the gray dashed line indicates the human expert performance, and the magenta dotted line indicates the Mondrian detection algorithm performance. The radii of the circles are proportional to the number of convolutional layers in the CNNs, indicated by the letters.

### TABLE VIII
SUMMARY OF KEY PERFORMANCE METRICS ON MUSCLE TEST SETS

<table>
<thead>
<tr>
<th>Test Set</th>
<th>Method</th>
<th>AUC</th>
<th>AU(PR)</th>
<th>At $\tau = 0.5$ Threshold</th>
<th>FA / Image</th>
<th>Precision</th>
<th>Recall</th>
<th>$F_1$</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>NSM1</td>
<td>Mondrian</td>
<td>0.892</td>
<td>0.111</td>
<td>1.472 0.135 0.513 0.214</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Human Expert</td>
<td>0.992</td>
<td>0.843</td>
<td>0.155 0.690 0.770 0.728</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>CNN $\mathcal{E}(A-H)$</td>
<td>0.985</td>
<td>0.756</td>
<td>0.298 0.545 0.796 0.647</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAN2</td>
<td>Mondrian</td>
<td>0.928</td>
<td>0.517</td>
<td>0.688 0.396 0.715 0.510</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Human Expert</td>
<td>0.983</td>
<td>0.802</td>
<td>0.161 0.759 0.723 0.751</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>CNN $\mathcal{E}(A-H)$</td>
<td>0.990</td>
<td>0.871</td>
<td>0.384 0.590 0.876 0.705</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TJM1</td>
<td>Mondrian</td>
<td>0.955</td>
<td>0.742</td>
<td>0.167 0.778 0.801 0.789</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Human Expert</td>
<td>0.998</td>
<td>0.991</td>
<td>0.010 0.985 0.902 0.942</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>CNN $\mathcal{E}(A-H)$</td>
<td>0.999</td>
<td>0.995</td>
<td>0.022 0.970 0.970 0.970</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ONM1</td>
<td>Mondrian</td>
<td>0.851</td>
<td>0.791</td>
<td>0.064 0.800 0.320 0.457</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Human Expert</td>
<td>0.976</td>
<td>0.974</td>
<td>0.011 0.984 0.840 0.906</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>CNN $\mathcal{E}(A-H)$</td>
<td>0.991</td>
<td>0.990</td>
<td>0.035 0.935 0.960 0.947</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GAM1</td>
<td>Mondrian</td>
<td>0.978</td>
<td>0.957</td>
<td>0.029 0.941 0.853 0.895</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Human Expert</td>
<td>0.997</td>
<td>0.992</td>
<td>0.014 0.972 0.920 0.945</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>CNN $\mathcal{E}(A-H)$</td>
<td>0.998</td>
<td>0.996</td>
<td>0.043 0.926 1.000 0.962</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
D. Additional Analysis

Possessing the prediction results of the human expert enables us to examine the predictions of the CNNs in a novel manner. For example, one can inspect the images on which the human and the CNNs agree or disagree to assess whether the CNNs’ predictions are understandable or unreasonable. To this end, a random set of example SAS images from each of the five test data sets is shown in Fig. 12 for the different combinations of human-expert predictions (denoted \(y_H\)) and CNN ensemble predictions (denoted \(y_E\)) made, when the true label is \(y\). (A decision threshold of \(\tau = 0.5\) was again used for the ensemble.) Additionally, for each test data set, the images for which maximum confusion occurred, meaning the ensemble of individual CNN predictions \((\mathcal{E})\) were in maximal disagreement, or the human-expert \((\mathcal{H})\) decision time was maximum, are also shown. For this assessment, we define the prediction confusion, \(\kappa\), on an image for a set of CNNs, \(\Omega\), with cardinality \(|\Omega|\), as

\[
\kappa(x|\Omega) = \left(|\Omega|(|\Omega| - 1)\right)^{-1} \sum_{i,j \in \Omega, i \neq j} \sqrt{(p(x|\omega_i) - p(x|\omega_j))^2},
\]  

(13)

where \(p(x|\omega_i)\) is the prediction of the \(i\)th CNN, \(\omega_i\), that the image \(x\) is a target.

From Fig. 12, the challenging nature of the data in terms of image quality and environmental diversity should be readily apparent. In particular, it can be seen that one cannot expect targets to possess the pristine signatures on which many shallow feature-based classifiers rely [60], [61] and which many synthetic data-modeling approaches produce [62]. Although different CNN architectures implicitly exploit different clues in the imagery when making predictions, a review of the classification scores confirms that the predictions of the CNN ensemble are quite robust and very rarely unreasonable. This broader insight is reinforced anecdotally by the images in Fig. 12 that exhibit high CNN confusion.

Next, the fractions of actual targets and clutter, across all five MUSCLE test data sets, for which the classification of the human expert and the ensemble of CNNs (at a decision threshold of \(\tau = 0.5\)) agreed or disagreed are summarized in Tables IX-X.

From the tables, it can be observed that the human-expert and the ensemble of CNNs make the same prediction on the vast majority of images. The most common case of disagreement is with targets that the CNN ensemble classifies correctly but that the human-expert does not; the human-expert was less conservative about mistaking targets for clutter.

But to more deeply analyze the results, we examine the actual images on which the two methods disagree. Interestingly, we find that there are consistent characteristics contained in the images on which the different permutations of disagreements arise. The clutter cases that the human-expert misclassifies as targets (but that the CNN ensemble classifies correctly) are almost always objects that are distinctly man-made, but that happen to not be of the target class (cf. Fig. 12, column 6, row 2). In contrast, the clutter cases that the CNN ensemble misclassifies (but that the human classifies correctly) usually contain the classical signature of a strong highlight followed (in range) by a strong shadow; however, the highlights in these cases may assume different forms, such as elongated bands (from sand ripple mounds), amorphous shapes, or linear features (cf. Fig. 12, column 7, row 5). The target cases that the human-expert misclassifies (but that the CNN ensemble classifies correctly) almost always are characterized by severely poor image quality; these images are blurry and defocused, with very ill-defined highlight structure (cf. Fig. 12, column 3, row 1). In contrast, the target cases that the CNN ensemble misclassifies (but that the human classifies correctly) tend to correspond to objects with weak, subtle highlights that merge with ripple highlights or other prominent background structure (cf. Fig. 12, column 2, row 2).

That is, the human has a more difficult time recognizing targets in degraded imagery, whereas the CNNs have a more difficult time when unusual seafloor or background characteristics interfere with the target signature. Conversely, the CNNs seem to implicitly understand the “essence” of a target, and how that fundamental nature can deform with imaging challenges, such as phase errors [63]. These insights suggest that exploiting auxiliary information derived from seafloor characterization [64], [65] and image-quality estimation [66], [67] might be fruitful avenues for improving classification performance further as in [68], [69].
Fig. 12. Example images from each MUSCLE test data set for the different combinations of human-expert predictions (denoted $y_H$) and CNN ensemble predictions (denoted $y_E$) made, when the true label is $y$. Each row corresponds to a unique test set, noted along the left side. Each of the first eight columns corresponds to a specific set of conditions, which appear above the top row. (A blank matrix entry indicates no images satisfied the specified condition.) The final four columns show the images for which maximum confusion occurred, meaning the ensemble of individual CNN predictions ($E$) were in maximal disagreement, or the human-expert ($H$) decision time was maximum. For each image, the (continuous-valued) prediction score of the CNN ensemble is also shown in the upper-left corner.
In most SAS images, objects are easily isolated from the background, and certainly more easily than in natural photographs (e.g., ImageNet images). In general, the background in SAS images is more uniform and less confusing – i.e., objects are more easily segmented out – with one important exception being seafloor characterized by sand ripples. As a result, only a modest number of (early) layers of a CNN would typically be needed to remove the sort of background information – extraneous to the classification task – that detection and segmentation processes address. Moreover, given the imaging geometry of the underwater application, the general shape of objects varies much less than in natural photographs. For example, the object scale does not depend on range in SAS imaging. Taken together, these elements suggest that SAS images are simply less complex than common natural images, and thus amenable to smaller CNNs.

To quantify this in a rigorous information-theoretic sense, we employ the compositional complexity of an image [70]. This measure has been shown to express the spatial heterogeneity of an image for a given partition. We calculate this measure of complexity for all of the MUSCLE SAS images and also for a subset of $1.2 \times 10^7$ ImageNet images. Specifically, we compute the Jensen-Shannon divergence for a set of partitions that uniformly divide the images into $\rho \in \{1, 2, 4, 8, 16, 32\}$ regions in each dimension. To enable fair head-to-head comparisons using this measure, the RGB-channel ImageNet images are first subjected to pre-processing steps that result in grayscale (i.e., luminosity) images of the same dimensions as the SAS images. Fig. 13 shows the resulting Jensen-Shannon divergence for the two image databases, confirming what can easily be seen visually: that the SAS images are substantially less complex, and that this holds across each scale (i.e., number of partition regions) considered. This result provides important justification for why tiny CNNs like the ones designed in this work are capable of succeeding.

Because the SAS images are less complex, it raises a question about what clues the CNNs are relying on to perform classification, and specifically whether the high resolution of the imagery is actually necessary. To investigate this idea, we intentionally degrade the resolution of one particularly interesting image chip, shown in Fig. 12’s ninth column and first row. The resolution is artificially decreased by applying an average pooling operation over blocks of size $\phi \in \{1, 2, 4, 8, 16\}$ pixels in each dimension (and then upsampling to revert to the original image size). The result of this operation is shown in Fig. 14, where $\phi = 1$ corresponds to the original full-resolution image, and the $\phi = 16$ case simulates an image with $24 \text{ cm} \times 24 \text{ cm}$ resolution. It can be seen that this process effectively smooths the image, progressively eliminating fine details as $\phi$ is increased.

Each of these reduced-resolution replicas is then submitted to the eight CNNs. At each intermediate layer of a given CNN, the difference of a replica’s response from the original ($\phi = 1$) image’s corresponding response is formed, and the mean value – averaged over the pixels in a map, and also the maps in the layer – is computed. This mean response difference is plotted in Fig. 15 for each layer of each CNN for the pooling factors $\phi$ considered with the image chip under study. In the legend, $\Delta$ corresponds to the difference at the output layer (i.e., in the final class prediction).

Several interesting trends can be noted from this figure. First, even with the most aggressive smoothing, $\phi = 16$, the final class prediction remains virtually unchanged for CNNs A, B, C, and D. Additionally, the smoothing does not significantly impact the prediction with CNN E up to a factor $\phi = 4$. In contrast, CNNs F, G, and H are severely impacted even when only a minimal reduction in resolution is applied. These results suggest which CNNs are exploiting the high-resolution details of the imagery – CNNs C, E, and F – and which are not. This knowledge can be leveraged in transfer-learning scenarios: the latter group of CNNs is likely to be more apropos for situations in which they are to be transferred to data from a lower-resolution sensor. These same CNNs are also likely to be more robust to minor imaging variations. The figure also hints at the stages within a given CNN where finer details start to be exploited, namely where the response difference grows. We hypothesize that the early layers of a CNN where the response difference remains low are likely performing simple smoothing operations rather than transformations that materially alter the object’s appearance.

To visually illustrate how significantly different clues are exploited by the different CNNs, we present in Appendix C the intermediate responses at each layer of the eight CNNs for the full-resolution image chip shown in Fig. 14.
Fig. 14. An example image chip and replicas whose resolution is artificially decreased by a factor $\phi$ in each dimension.

Fig. 15. At each intermediate layer of a given CNN, the mean response difference between the original full-resolution ($\phi = 1$) image in Fig. 14 and a replica with resolution decreased by a factor $\phi$ in each dimension. The layer type abbreviations correspond to input (I), convolutional layer (C), pooling layer (P), dense layer (D), and output (O). In the legends, $\Delta$ corresponds to the difference at the output layer (i.e., in the final class prediction).

The fact that the SAS target classification task considered here is binary in nature – in contrast to the polychotomous problem addressed with ImageNet images – is yet another reason that tiny CNNs are likely to succeed. With only two classes to consider, the task of learning appropriate class boundaries is necessarily simpler for the CNN than in a multi-class problem [71], [72]. And in turn, a “thin” (and shallow) CNN that relies on only a limited number of convolutional filters should be adequate. The necessity of deep networks arises principally due to images that are particularly challenging. We underscore this point by examining the behavior of CNNs in the context of images that are “shallow-learnable” – i.e., capable of being
correctly classified by a shallow classifier – as done in [73], for three CNNs that span the range of network depths considered in our work.

Fig. 16 shows the evolution of the CNNs’ AUC on subsets of test images formed based on whether or not the shallow (Mondrian) algorithm had classified the images successfully, as well as the ratio of those AUC values. (By construction, the corresponding AUC of the Mondrian algorithm on the two subsets are 1 and 0.) From the figure, it can be seen that the networks correctly classify first the shallow-learnable images, and that the increased filter specificity that occurs with more epochs of training is needed only for the small fraction (here, about 5%) of remaining non-shallow-learnable images. (Interestingly, CNN E, which was shown in Fig. 15 to exploit high-resolution details, performs better on the challenging subset than the other two CNNs that did not leverage such information.) This behavior of the CNNs – and in particular of the smallest, CNN A – further helps to explain why tiny networks can be successful in sonar classification tasks.

V. Conclusion

A novel approach to the design of CNNs for underwater target classification tasks with SAS imagery was proposed. The careful design of the CNN architectures dramatically reduces the number of free parameters to be learned by orders of magnitude. This in turn eases the training data requirements while also greatly accelerating the CNN training phase. Experimental results on multiple challenging data sets collected at sea demonstrated that these proposed CNNs can attain and even surpass the performance achievable by a human domain-expert.

There are four key components to our approach of using tiny CNNs to achieve super-human target classification performance in SAS imagery: (i) efficient CNN architectures with limited numbers of filters but many convolutional layers; (ii) on-the-fly data augmentation in the training phase, especially to compensate for limited target views; (iii) the use of a set of isometric inputs in the evaluation stage to improve robustness; and (iv) pooling predictions across an ensemble of unique CNNs to enable the exploitation of clues from different intermediate CNN representations. We believe these findings can significantly impact the manner in which CNNs are utilized in the underwater remote-sensing community. To wit, the tiny CNNs proposed here provide a blueprint for achieving excellent classification performance even with limited computing power or limited data.

These tiny CNNs generalize well to markedly distinct test data sets, but they can also be amenable for exploitation in a transfer-learning paradigm to similar, but different, sensors. Preliminary results from ongoing work provide evidence that these CNNs can indeed be very useful as intelligent initializations for performing classification on sensor data for which limited training data is available. Anecdotally, we have also applied these trained CNNs to a handful of images from other SAS sensors – directly, without any additional refinement – and the classification results “out of the box” are quite promising.

Future work will more fully explore the benefits and limitations of the transfer-learning concept to other SAS and side-scan sonar sensors. Other work that is ongoing is the application of the trained CNNs directly to scene-level imagery as a unified, single-stage detector and classifier [74], and the use of generative adversarial networks (GANs) to augment the data sets with synthetically generated target imagery [75]. And finally, we hope to compare the performance on these data sets of our tiny CNNs to more typical (million-plus parameter) CNNs that other groups are designing.

Acknowledgments

The author gratefully acknowledges the efforts of the 12 anonymous human subjects who participated in the labeling experiment.

This work was supported by the NATO Allied Command Transformation.
APPENDIX A

For a general idea of what sorts of filters these tiny CNNs learn, the convolutional filters (without the bias terms) of CNN E when trained on the MUSCLE data are shown in Fig. 17.

APPENDIX B

Performance of the 12 human subjects on the ONM1 test set is shown in terms of full ROC-like curves and precision-recall curves in Fig. 18. The AUC (for the corresponding ROC curve) and AU(PR) are also provided in the legends. On each curve, the operating point corresponding to the result of the simple binary labeling (and thus independent of time) is also marked; this point can also be viewed as representing the standard decision threshold of $\tau = 0.5$. Also shown is the ensemble result when averaging across all 12 human subjects' binary predictions (i.e., ignoring the time component, to avoid different baseline temporal biases across subjects); specifically, in this case, the prediction for a given image is taken to be the mean of the humans' 12 assigned labels each in \{0, 1\}.

A summary of the human subjects' performance on the ONM1 test set is shown in Table XI. Based on the AUC and AU(PR) scores, subject 6 performed best; but the best-performing subject who was available to further label the entire collective test data was subject 9, the runner-up. Therefore this latter subject, who did achieve comparable performance, was selected as the human expert, and all results for the “Human Expert” correspond to this individual. Interestingly, the ensemble of humans achieved a higher AUC than any single human. However, even that case performed worse than almost every individual CNN, as well as the CNN ensemble.

APPENDIX C

To illustrate how significantly different clues are exploited by the different CNNs, we present the intermediate responses at each layer of the eight CNNs for one interesting image chip. Specifically, we consider the image chip shown in Fig. 12’s ninth column and first row, which was the target in the NSM1 test data set for which the CNNs disagreed maximally; it is reproduced here for convenience in Fig. 19.

The intermediate responses at each layer of the 8 CNNs (trained on MUSCLE data) for the image in Fig. 19 are shown in Figs. 20-27. Interestingly, some of the CNNs effectively mask the non-uniform background response, while others do not. The variation in response between two CNNs with the same depth, but different filter sizes and pooling factors, can also be observed, for instance, in Fig. 24 and Fig. 25.

Examining intermediate responses like these can aid in the interpretation of what clues the filters are exploiting. From the figures, it can be seen that many layers within some of the CNNs perform only subtle transformations that resemble common operations like despeckling, background removal, and intensity normalization. The level of abstraction, to representations that are no longer easily associated with the original input imagery, often increases only deeper in the networks. For example, in Fig. 24, the relationship of the input image to the intermediate representations produced after 10 (convolution and pooling) layers of CNN E is still visible.
Fig. 17. The convolutional filters learned for CNN E using MUSCLE training data. 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). For the 3-d tensors, filter depth runs vertically down the page.
Fig. 18. Human-subjects’ performance on the ONM1 test data set, in terms of (a) ROC-like curves and (b) precision-recall curves. The operating point equivalent to a $\tau = 0.5$ threshold is marked with a solid dot. In (a), the AUC for the corresponding ROC curve is given in the legend. In (b), $A$ in the legend indicates the area under the precision-recall curve.

<table>
<thead>
<tr>
<th>Subject</th>
<th>AUC</th>
<th>AU(PR)</th>
<th>FA / Image</th>
<th>Precision</th>
<th>Recall</th>
<th>$F_1$ Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human 1</td>
<td>0.919</td>
<td>0.902</td>
<td>0.011</td>
<td>0.976</td>
<td>0.533</td>
<td>0.690</td>
</tr>
<tr>
<td>Human 2</td>
<td>0.962</td>
<td>0.950</td>
<td>0.043</td>
<td>0.944</td>
<td>0.893</td>
<td>0.917</td>
</tr>
<tr>
<td>Human 3</td>
<td>0.909</td>
<td>0.865</td>
<td>0.223</td>
<td>0.767</td>
<td>0.920</td>
<td>0.836</td>
</tr>
<tr>
<td>Human 4</td>
<td>0.949</td>
<td>0.931</td>
<td>0.053</td>
<td>0.925</td>
<td>0.826</td>
<td>0.873</td>
</tr>
<tr>
<td>Human 5</td>
<td>0.910</td>
<td>0.848</td>
<td>0.064</td>
<td>0.887</td>
<td>0.627</td>
<td>0.734</td>
</tr>
<tr>
<td>Human 6</td>
<td>0.981</td>
<td>0.979</td>
<td>0.021</td>
<td>0.972</td>
<td>0.920</td>
<td>0.945</td>
</tr>
<tr>
<td>Human 7</td>
<td>0.891</td>
<td>0.876</td>
<td>0.011</td>
<td>0.984</td>
<td>0.840</td>
<td>0.906</td>
</tr>
<tr>
<td>Human 8</td>
<td>0.946</td>
<td>0.898</td>
<td>0.085</td>
<td>0.895</td>
<td>0.907</td>
<td>0.901</td>
</tr>
<tr>
<td>Human 9</td>
<td>0.976</td>
<td>0.974</td>
<td>0.111</td>
<td>0.800</td>
<td>0.907</td>
<td>0.850</td>
</tr>
<tr>
<td>Human 10</td>
<td>0.937</td>
<td>0.897</td>
<td>0.181</td>
<td>0.880</td>
<td>0.907</td>
<td>0.850</td>
</tr>
<tr>
<td>Human 11</td>
<td>0.947</td>
<td>0.929</td>
<td>0.117</td>
<td>0.880</td>
<td>0.907</td>
<td>0.868</td>
</tr>
<tr>
<td>Human 12</td>
<td>0.939</td>
<td>0.936</td>
<td>0.011</td>
<td>0.985</td>
<td>0.867</td>
<td>0.922</td>
</tr>
<tr>
<td>Human Ensemble</td>
<td>0.988</td>
<td>0.984</td>
<td>0.011</td>
<td>0.985</td>
<td>0.867</td>
<td>0.922</td>
</tr>
</tbody>
</table>
Fig. 20. For the image chip in Fig. 19, the intermediate responses at each layer of CNN A. Each row corresponds to the output at one layer (convolutional or pooling) of the CNN.

Fig. 21. For the image chip in Fig. 19, the intermediate responses at each layer of CNN B. Each row corresponds to the output at one layer (convolutional or pooling) of the CNN.
Fig. 22. For the image chip in Fig. 19, the intermediate responses at each layer of CNN C. Each row corresponds to the output at one layer (convolutional or pooling) of the CNN.

Fig. 23. For the image chip in Fig. 19, the intermediate responses at each layer of CNN D. Each row corresponds to the output at one layer (convolutional or pooling) of the CNN.
Fig. 24. For the image chip in Fig. 19, the intermediate responses at each layer of CNN E. Each row corresponds to the output at one layer (convolutional or pooling) of the CNN.

Fig. 25. For the image chip in Fig. 19, the intermediate responses at each layer of CNN F. Each row corresponds to the output at one layer (convolutional or pooling) of the CNN.
Fig. 26. For the image chip in Fig. 19, the intermediate responses at each layer of CNN G. Each row corresponds to the output at one layer (convolutional or pooling) of the CNN.

Fig. 27. For the image chip in Fig. 19, the intermediate responses at each layer of CNN H. Each row corresponds to the output at one layer (convolutional or pooling) of the CNN.

D. Williams, “Fast unsupervised seafloor characterization in sonar...”.


