2017 IEEE 56th Annual Conference on Decision and Control (CDC)
December 12-15, 2017, Melbourne, Australia

Deep Learning Feature Extraction for Target Recognition and Classification in Underwater Sonar Images

Pingping Zhu[1], *Member, IEEE,* Jason Isaacs[2], Bo Fu[3], *Member, IEEE,* and Silvia Ferrari[4], *Senior Member, IEEE*

Abstract— **This paper presents an automatic target recognition (ATR) approach for sonar onboard unmanned underwater vehicles (UUVs). In this approach, target features are extracted by a convolutional neural network (CNN) operating on sonar images, and then classified by a support vector machine (SMV) that is trained based on manually labeled data. The proposed approach is tested on a set of sonar images obtained by a UUV equipped with side-scan sonar. Automatic target recognition is achieved through the use of matched filters, while target classification is achieved with the trained SVM classifier based on features extracted by the CNN. The results show that deep learning feature extraction provide better performance compared to using other feature extraction techniques such as histogram of oriented gradients (HOG) and local binary pattern (LBP). By processing images autonomously, the proposed approach can be combined with onboard planning and control systems to develop autonomous UUVs able to search for underwater targets without human intervention.**

I. Introduction

Automatic target recognition (ATR) and classification are important for a wide range of autonomous systems and applications. In modern maritime operations, vehicles outfitted with acoustic sensors, such as side-scan sonars, are used to obtain images of unidentified objects that may be of potential threat [1], [2]. Unmanned underwater vehicles (UUVs) are particularly suited for this task, and generally, they are guided to take images based on prior available surveying and tactical information. ATR eliminates the need of manually classifying targets by expert human operators, which can be costly, slow and inefficient. ATR based on underwater sonar images faces challenges because sonar images are natural images characterized by low constrast, low resolution, and noise and clutter. As a result, many handcrafted features used in computer vision are unlikely to extract meaningful information. This paper leverages recent advancements in deep learning combined with matched filters and classifier to extract dominant features from underwater sonar images for ATR. This is an important stepping stone for the development of sonar-driven path planning for autonomous UUVs.

The recent success of deep learning algorithms for object recognition in images is due to the ability to effectively perform highly nonlinear feature extraction. [3]–[11]. Convolutional neural networks (CNNs) are one of the most effective deep learning architecture used for image feature extraction that have spurred a rapid improvement in visual recognition and brought forth dramatically improved performance. Such improvements have been demonstrated through the yearly ImageNet Large-Scale Visual Recognition Challenge (ILSVRC), which is designed to allow researchers to compare progress in computer vision [12]. This paper shows that by using the renowned robust pre-trained CNN network *AlexNet*, trained with ordinary images [7], in combination with a matched filter and a support vector machine (SVM) classifier, deep learning can be applied to feature extraction in pre-processed underwater side-scan sonar images, without having to train a new CNN purely on sonar domain.

The rest of the paper is organized as follows. In Section II, problem formulation is presented. In Section III the approach used for object recognition in sonar images is described, as well as other topics such as sonar image pre-processing and object recognition based on the matched filters. In Section IV, background on CNN is presented, followed by a description of the object classification system in Section V-B. Results are presented and discussed in Section VI. Finally, conclusion and future directions are given in Section VII.

II. Problem Formulation

Consider the problem in which multiple images are obtained by a mobile UUV, to recognized, segment, and classify one or more objects of interest, each belonging to one of two classes referred to as *target* c_0 and *non-target* c_1. The goal of the classification task is to design a classifier, f, which maps an $(n_s' \times n_t')$ image segment matrix K to the output $y \in \{0, 1\}$. Then, the classifier is applied to determine the class $u \in \{c_0, c_1\}$ from image segments based on the following decision policy:

$$u = \begin{cases} c_0, & \text{if } y = f(K) = 0 \\ c_1, & \text{if } y = f(K) = 1. \end{cases} \tag{1}$$

The image segment matrix K is obtained by segmentation of the $(n_s \times n_t)$ seafloor image matrix I. Each element of the matrices I and K has a non-negative value,

$$I(i, j),\ K(\iota, \zeta) \in [0, +\infty). \tag{2}$$

Let (i_I, j_I) be a set of indices of the image matrix I. Choosing the first element of K to be at the location of

[1]Mechanical and Aerospace Engineering, Cornell University, Ithaca, NY, US pingping.zhu@cornell.edu
[2]Naval Surface Warfare Center, Panama City Division, Panama City, FL, US jason.c.isaacs1@navy.mil
[3]Mechanical and Aerospace Engineering, Cornell University, Ithaca, NY, US bf284@cornell.edu
[4]Mechanical and Aerospace Engineering, Cornell University, Ithaca, NY, US ferrari@cornell.edu

$I(i_I, j_I)$, the relationship between K and I can be expressed explicitly as

$$I = \left[\begin{array}{ccccc} & & \vdots & & \\ \underbrace{\cdots \quad \cdots}_{j_I - 1} & & \underbrace{K}_{n'_t} & & \underbrace{\cdots \quad \cdots}_{n_t - n'_t + i_I + 1} \\ & & \vdots & & \end{array} \right] \begin{array}{l} \} \, i_I - 1 \\ \} \, n_s \\ \} \, n_s - n'_s - i_I + 1 \end{array} \tag{3}$$

Since K always lies inside the image matrix I, the location of the first pixel of K and the indices of K are constrained by $i_I \in [1, n_s]$; $j_I \in [1, n_t]$; and $\iota \in [1, n_s + 1 - i_I]$; $\iota \in [1, n_t - j_I]$, respectively.

In this paper, underwater sonar images refer to images of the seafloor taken by a moving UUV equipped with side-scan sonar, as shown in Fig. 1. The sonar is mounted on the bottom right and bottom left of the UUV, and emits acoustic waves directly below the UUV. The acoustic waves are emitted at time intervals one Δt apart, and the reflected waves are recorded. The UUV moves forward (defined as parallel to the seafloor) with constant velocity v. During the jth time intervals, sonar reflection data from a scan is recorded, as shown by the red solid lines in Fig. 1. These data are stored in the jth column of I. The actual size of the sea floor represented by one sonar image is determined by the sonar range L_r and the total travel distance $L = n_t v \Delta t$. Each pixel value $I(i, j)$, represents the strength of the reflected acoustic waves from the sea floor. The image matrices obtained by the right and left side sonars are denoted by I_L and I_R, respectively, as shown in Fig. 1. However, since all data processing treatments are identical for the left and right images, the subscripts L and R will be omitted for simplicity, and I will be used to refer to either the right or the left side sonar scan data.

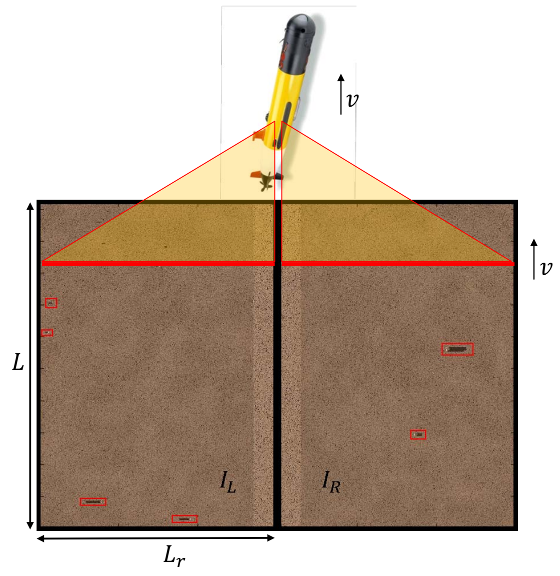


Fig. 1. UUV and side-scan sonar image data schematics

To collect the underwater images, the simulated UUV navigates near the sea floor according to a given trajectory, as shown by the green curve in Fig. 2. In this Figure, the green trajectory is the coverage path (similar to that the zig-zag motion for a lawnmower). Objects of potential interest are defined to be either targets (shown by blue circles) or non-targets (shown by red dots). A total of $N = 35$ sonar images are obtained along the given UUV trajectory (Fig. 2). The corresponding locations of these images are shown by red numbers from 1 to 35. Image data at different locations along travel trajectory are identified using superscript n, such that the image at the nth location is $I^{(n)}$, where $n = 1, ..., N$. The problem considered in this paper is to recognize and classify targets in the sonar images obtained by the UUV along a given trajectory. This is accomplished by first recognizing objects of interest from the sonar images, and then classifying the objects recognized into a class u. In particular, given a sonar image, $I(i, j)$ for $i = 1, ..., n_s$ and $j = 1, ..., n_t$, an object of interest is recognized by finding a segment represented by a sub-matrix K in the given sonar image matrix I.

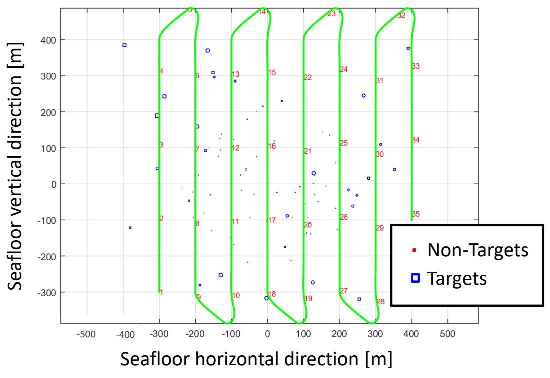


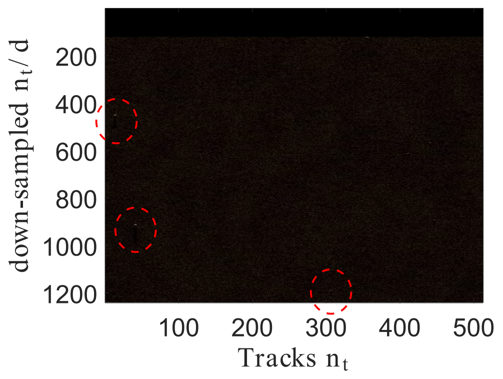
Fig. 2. UUV trajectory, sonar image location and target location

III. Object Recognition

In this section, the recognition of object of interest in sonar images is described. The histogram equalization technique is used in the image pre-processing phase. A matched filter is designed for the seafloor sonar image objects, and used later for image recognition. The recognized object images are then considered as input of the proposed classifier in Section V-B.

A. Image Pre-processing

Because the matrix I is a record of the reflected acoustic wave strength, it is not a standard image matrix. Pre-processing is necessary to enhance the measurements for further processing. Also, since the original sonar data are over-sampled, the first step in pre-processing is to down-sample sonar matrix data by a user-defined factor, d. This reduces the computational complexity of the problem. Then, the down-sampled image I_d becomes a matrix of size of $\lfloor \frac{n_s}{d} \rfloor \times n_t$, where $\lfloor \cdot \rfloor$ is the floor operator. Next, the down-sampled image is normalized linearly to obtain the grayscale



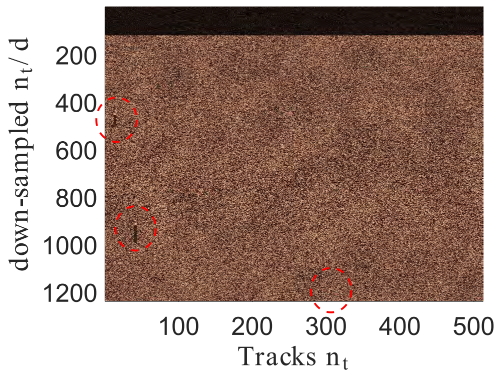


Fig. 3. Gray image I_g (upper) and histogram equalized image I_h (lower). Objects are circled in red.

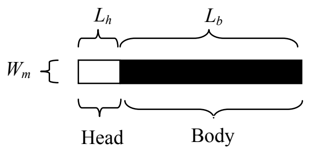


Fig. 4. Matched filter with head and body components.

image matrix,

$$I_g(i,j) = \frac{I_d(i,j)}{\max\limits_{i,j}\left[I_d(i,j)\right]} \tag{4}$$

Then $I_g(i,j) \in [0,1]$, $\forall i,j$. Finally, in order to adjust image intensities to enhance contrast, the histogram equalization technique [13] is applied to the gray image, and a transformed image, I_h, is obtained. Examples of the grayscale image I_g and the histogram equalized image I_h are shown in Fig. 3. It can be seen that the objects in I_h are better observed I_h than in I_g due to enhanced contrast.

B. Segmentation based on Matched Filters

In sonar images used in this study, all objects of interest have a similar structure comprised of a highlight area followed by a shadow. This is because the object reflects the sonar waves causing a the sonar to pick up a strong signal for that location, while location behind the object is blocked, results in a weak signal registration. The direction of the shadow area is always in line with the sonar scan direction, facing away from the sonar. To recognize and segment seafloor objects in the sonar images, a matched filter is designed, as shown in Fig. 4. W_m is the width of the matched filter and L_h and L_b are the lengths of the head and body parts, respectively. The matched filter is mathematically described by introducing a $(W_m \times (L_h + L_b))$ match filter matrix,

$$I_m(i,j) = \begin{cases} 1, \text{for } i \in [1, W_m] \text{ and } j \in [1, L_h] \\ -1, \text{for } i \in [1, W_m] \text{ and } j \in [L_h+1, L_h+L_b] \end{cases} \tag{5}$$

The grayscale image I_g is then converted to the binary image matrix

$$I_b(i,j) = \begin{cases} 1, & \text{if } I_g(i,j) \geq \theta_b \\ -1, & \text{if } I_g(i,j) < \theta_b \end{cases} \tag{6}$$

where θ_b is the binary threshold for the pixels. The normalized output of the matched filter is expressed as

$$I_n(i,j) = \frac{\sum_{\iota=1}^{W_m} \sum_{\zeta=1}^{L_h+L_b} I_b(i+\iota, j+\zeta) I_m(\iota,\zeta)}{\sum_{\iota=1}^{W_m} \sum_{\zeta=1}^{L_h+L_b} I_m^2(\iota,\zeta)}, \tag{7}$$

and is plotted in Fig. 5. Large values of $I_n(i,j)$ (peaks in Fig. 5) indicate that the pixels around them agree with the designed matched filter. Let θ_m be a user selected threshold value. $I_n(i,j) \geq \theta_m$ is selected, and their indices are grouped into a set denoted by $\boldsymbol{\sigma} = \{(i,j) | I_n(i,j) \geq \theta_m\}$.

Shadow lengths of different objects varies in a sonar images. Also, shadow length of the same object varies from sonar image to sonar image, because the sonar orientations and ambient lighting conditions may have changed. Thus, k_0 matched filters $\{I_m(k)\}_{k=1}^{k_0}$, with different length parameters $\{L_b(k)\}_{k=1}^{k_0}$ and corresponding binary thresholds $\{\theta_b(k)\}_{k=1}^{k_0}$, are applied to recognize different objects in images. For the kth matched filter, points on an image selected by this matched filter can be represented by $\boldsymbol{\sigma}(k)$. Therefore the combined set of points selected by all k_0 matched filter is

$$\boldsymbol{\sigma}_{tot} = \boldsymbol{\sigma}(1) \cup ... \cup \boldsymbol{\sigma}(k) \cup ... \cup \boldsymbol{\sigma}(k_0). \tag{8}$$

After obtaining $\boldsymbol{\sigma}_{tot}$, the points of an image selected by all the matched filters are clustered into several 8-connected objects and identified in white as shown in Fig. 6. Finally, the image segment of the objects of interest K is obtained from the original sonar image I. Some examples of the image segments are shown in Fig. 7.

IV. Background on Convolutional Neural Networks

Convolutional neural networks (CNNs) are a modification of artificial neural networks (ANNs) that employs multiple heterogenous layers in a cascade structure, such as that shown in Fig 8, where each layer allows learning and feature extraction at different levels starting with a row and potentially complex image as the input. Though many CNN structures have been proposed, they all incorporate several

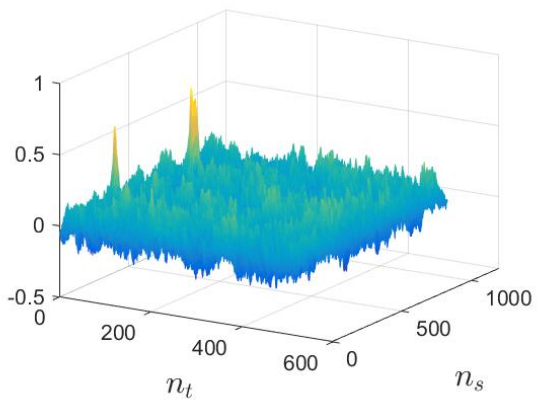


Fig. 5. The normalized output of the matched filter.

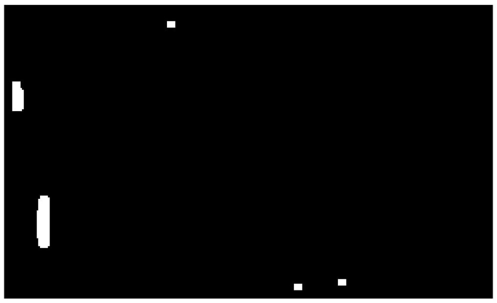


Fig. 6. Example σ set for a sonar image represented by white segments.

key layers, including the convolutional layer and Rectified Linear Units (ReLU), pooling, cross channel normalization, and conventional fully-connected (FC) layers, which are briefly described in the following subsections.

A. Convolutional layer and ReLU layer

The convolutional layer is of key importance to a CNN because it is where most of the feature extraction computation takes place. A CNN may have many convolutional layers. A general convolutional layer uses a $(n_{D1} \times n_{W1} \times n_{H1})$ 3-D volume $X(d, i, j)$ as input, and outputs a $(n_{D2} \times n_{W2} \times n_{H2})$ 3-D volume $O(k, i, j)$. The indicies i and j denote the column and row position of an element of an image matrix K, d is the depth of the image (i.e. $d = 3$ for RGB images), and k is the number of $(F \times F)$ convolutional kernels $\Omega(\iota, \zeta)$ used in the operation. A convolutional kernel has the effect of a filter on the input image.

In order to assure that the images before and after the convolution layer have the same width and height, the input

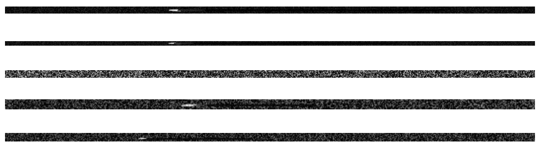


Fig. 7. Examples of recognized image segments rotated 90 degrees counterclockwise.

image is first padded with P layers of zeros, see Fig. 9. The convolutional kernel Ω is applied to the padded input image similar to that of an image filter, which slides at a rate of S pixel elements per operation, along each rows of the input image. S is also referred to as the *stride* parameter. Using P and S, the input and output dimensionalities of the convolutional layer are related by

$$\begin{aligned} n_{W2} &= (n_{W1} - F + 2P)/S + 1 \\ n_{H2} &= (n_{H1} - F + 2P)/S + 1. \end{aligned} \tag{9}$$

For the first convolutional layer, the padded segmented image matrix is used as an input, where the depth parameter n_{D1} is 1 for a single layer image. A more general example of a $n_{D1} = 3$ and $n_{D2} = 4$ convolutional layer is shown in Fig. 10, where the 4 different colors indicate the 4 different convolutional kernels.

Each element of a general convolutional layer output O is obtained via a convolutional operation

$$\begin{aligned} &O(k, i, j) \\ &= \sum_{d=1}^{n_{D1}} \sum_{\iota=1}^{F} \sum_{\zeta=1}^{F} \omega_{k,d,\iota,\zeta} X'\left(d, i(S-1)+\iota, j(S-1)+\zeta\right), \end{aligned} \tag{10}$$

where X' is a matrix that is the extension of the matrix X by zero-padding, and $\omega_{k,d,\iota,\zeta}$ is the weighting parameter of the (ι, ζ) element of the k'th kernel used in the d'th layer convolution. As an example, the convolutional operation for a $(1 \times 5 \times 5)$ input matrix X with $P = S = 1$ and a kernel size $F = 3$ is shown in Fig. 9. For example, the first element of the output matrix at the first convolution step $(i = j = 1)$ is computed as

$$O(1,1,1) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 2 \end{bmatrix} \circ \begin{bmatrix} 4 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 2 \end{bmatrix} = 7 \tag{11}$$

where $\circ$ denote the element-wise Hadamard product of two matrices.

Since the convolution operation is a linear operation, non-linearity needs to be introduced to the network so that the CNN can correctly capture the non-linear relationship between the input image and the output features. This can be done by introducing an activation layer that applies an element-wise activation function. The rectified linear unit, or ReLU layer is a layer that applies an element-wise non-saturating activation function $f(x) = \max(0, x)$. This activation function has been demonstrated to be much more computationally effective in CNN than the logistic sigmoid typically used in ANNs.

B. Cross Channel Normalization Layer

For some CNNs such as the AlexNet, local normalization is applied after the ReLU layer, and has been shown to reduce error rate [7]. The cross channel normalization layer is a local normalization scheme used in the AlexNet. Let $a^{\varsigma}(i, j)$ denote the activity of a neuron computed by first applying the ςth 2-D kernel at position (i, j) and then applying the

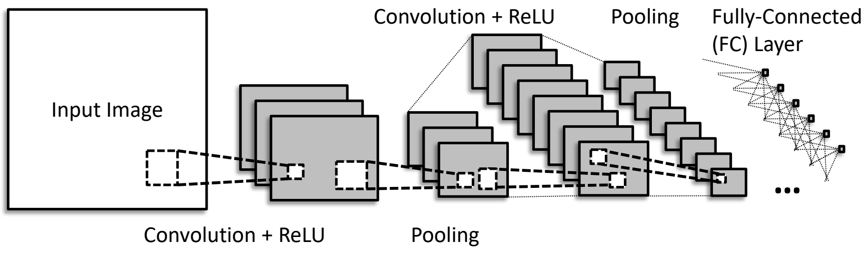


Fig. 8. Structure of a convolutional neural network

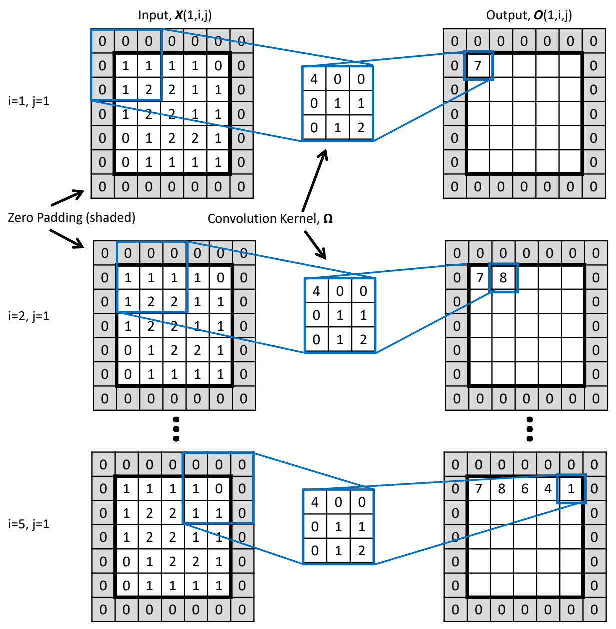


Fig. 9. Example of convolutional operation.

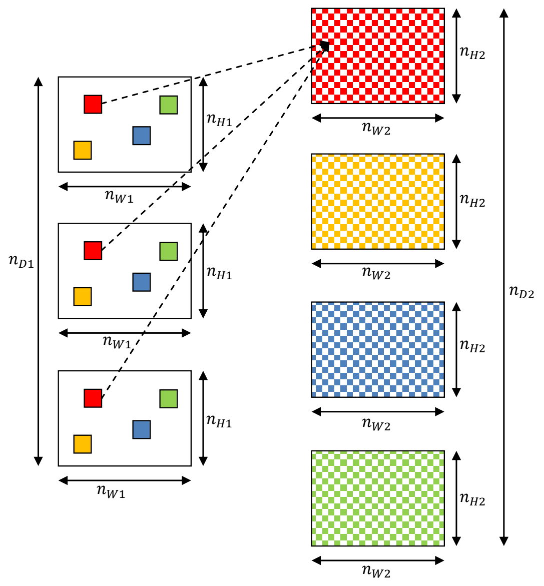


Fig. 10. Convolutional layer input and output dimensionality.

ReLU nonlinearity. The response-normalized activity $b^{\varsigma}(i,j)$ is given by

$$b^{\varsigma}(i,j) = \frac{a^{\varsigma}(i,j)}{\left\{q + \alpha \sum_{\tau=\tau_i}^{\tau_f} \left[a^{\varsigma}(i,j)\right]^2\right\}^{\beta}}, \quad (12)$$

$$\tau_i = \max(0, \varsigma - \ell/2) \quad (13)$$

$$\tau_f = \min(n_k - 1, \varsigma + \ell/2) \quad (14)$$

where the sum runs over ℓ "adjacent" kernel maps at the same spatial position, and n_k is the total number of kernels in the layer. The parameters q, α, l, β are considered as constant user-defined hyper-parameters, see [7] for more details.

C. Pooling Layer

The Pooling layer is a nonlinear down-sampling operation along the width and height of the input volume. The purpose of a pooling layer is to progressively reduce the size of the volume, leading to a reduction of the amount of parameters in the network. This then improves computation efficiency and also gives a way to control overfitting. The Pooling Layer operates independently on every depth slice of the input volume, outputing the downsampled value as the maximum of the input values. The most common form of a pooling layer is a 2×2 matrix applied with a stride $S = 2$ downsamples every depth slice in the input by 2, along both width and height. Example of a pooling operation on a 2×2 input matrix with the stride parameter $S = 2$ is shown in Fig. 11. Notice after the pooling operation, the size of the input volume has reduced by a factor of 4. In this example, the stride parameter is equal to the size of the size of the pooling operator, and the result is that the pooling layer input matrices do not overlap. However, S can be smaller than the size of the pooling operator, which results in overlapping pooling. In the AlexNet, overlapping pooling slight improved network performance. Although using a pooling layer for downsampling is still popular, other alternative, such as increased stride in the convolutional layer, have been proposed [14].

D. Fully-Connected (FC) Layers

After the network has down-sampled through a series of convolutional and pooling operations, fully-connected (FC) layers are then used to combine information from the last network layer to extract features. It can be loosely viewed as a 1 dimensional convolutional layer. Being one dimensional, the neurons are the same as those found in an ANN, and that

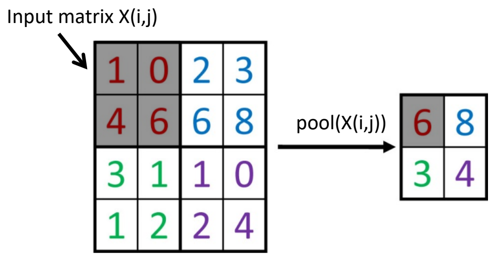


Fig. 11. Pooling operation for 2×2 input matrix. Stride $S = 2$.

any one is connected to all other neurons in the previous layer. Several FC layers can be used together to improve learning and prevent underfitting. The output of the final FC layer is a features vector $\mathbf{z}$.

V. Object Classification

Due to the nature of underwater sonar image data mentioned in Section I, direct training of a classifier using the sonar images will yield poor results. Thus, the image segments, K's in the previous section is used instead. Since these image segments cannot be applied directly to classification, the features of these image segments are first extracted using a pre-trained CNN, and then the image segments are classified based on extracted features.

A. Feature Extraction

The AlexNet is chosen as a pre-trained network to demonstrate how the limited availability of sonar images may be overcome by using CNNs trained on other images and domains. AlexNet is a robust network containing five convolutional and three fully-connected layers. It is trained using the data set ImageNet LSVRC-2010 that includes 1.2 million photographs from 1000 object categories. Details on the AlexNet can be found in [7]. The AlexNet, shown in Fig 12, uses the recognized image segments as inputs and extract its features, producing a (4096×1) features vector $\mathbf{z}$. Since the AlexNet is pre-trained, the network stays the same for both the target training phase and testing phase. In this study the features vector $\mathbf{z}$ is obtained from the sixth layer of the AlexNet, where the rest is replaced by a support vector machine (SVM).

B. Classification from Extracted Features

Because of the high dimension of the extracted features, a linear support vector machine (SVM) is applied to classify these image segments [15], which is expressed by

$$y = f_{SVM}(\mathbf{z}) = \Phi(\mathbf{w}^T\mathbf{z} + b) \tag{15}$$

where $\Phi(x) = 1$ if $x \leq 0$, and $\Phi(x) = 0$ otherwise, which denotes a mapping from $(\mathbf{w}^T\mathbf{z}+b) \in \mathbb{R}$ to the class label $y \in \{0, 1\}$. This parameter $\mathbf{w}$ can be learned from the training data set by solving the following optimization problem

$$\begin{aligned} &\min_{\mathbf{w},b,\boldsymbol{\xi}} \tfrac{1}{2}\|\mathbf{w}\|^2 + c\textstyle\sum_{n=1}^{N_{tr}} \xi_n \\ s.t. \quad &\xi_n \geq 0, \quad y_n(\mathbf{w}^T\mathbf{z}_n + b) \geq 1 - \xi_n, \\ &n = 1, ..., N_{tr} \end{aligned} \tag{16}$$

where $\boldsymbol{\xi} = [\xi_1, ..., \xi_{N_{tr}}]^T$ are the slack variables. They represent the degree that each data sample lies inside the margin, defined by two hyperplanes, $(\mathbf{w}^T\mathbf{z} + b) = \pm 1$. The user-defined parameter $c > 0$ controls the trade-off between the slack variable penalty and the margin [16]. Here, the training data set is $\mathcal{D} = \{(\mathbf{z}_n, y_n)\}_{n=1}^{N_{tr}}$ and N_{tr} is the number of training data. In this paper, the training data set is obtained from the original sonar image manually.

VI. Simulations and Results

The proposed ATR approach is demonstrated using $N = 35$ sonar images obtained by the UUV shown in Fig. 2, following the testing procedure outlined in Fig. 12, where images are used to produce a training and a testing set. In the training phase, the image segments in the training set are recognized and segmented manually, and the image segments representing the objects of interest are manually labeled based on the ground truth. Then, these image segments are feed into the AlexNet to extract salient features, as described in Section V-A. Finally, the extracted features (outputs of the 'fc6' layer) and the target labels are applied to train the SVM, as described in Section V-B. The feature layer ('fc6' layer) is selected based on comparison of classification performance among all FC layers.

In the testing phase, the image segments are recognized and segmented automatically using the image processing method presented in Section III. Similarly, the recognized image segments are feed into the AlexNet for features extraction. Finally, the extracted features are applied as inputs of the trained SVM, and the outputs of the SVM is the predicted class of the corresponding image segments. The cross validation method is used to generate additional training and testing data sets. From all of the sonar images, $n_{tr} = 233$ image segments are recognized and segmented manually. They are all used as training image segments. First, sonar images are split into 5 Groups denoted by $\mathcal{G}_1$, $\mathcal{G}_2$, $\mathcal{G}_3$, $\mathcal{G}_4$, and $\mathcal{G}_5$ according to Table II. There are about 50 training image segments in each group. Denote the set of all index by $\mathcal{I} = \{1, 2, 3, 4, 5\}$. Each time, one group denoted by $\mathcal{G}_t$, $t \in \mathcal{I}$, is applied as test image set and the other four groups denoted by $\mathcal{G}_0 = \bigcup_{t' \neq t, t' \in \mathcal{I}} \mathcal{G}_{t'}$ are applied as training image set. Then, the image segments obtained manually from the sonar image $I_0 \in \mathcal{G}_0$ are applied as training data set. Finally, the image segments recognized and segmented automatically from the sonar image $I_t \in \mathcal{G}_t$ are applied as the testing data set.

To performance of the proposed deep learning approach is evaluated by comparing its target classification performance to two other existing methods, the Local binary patterns (LBP) features [17] and the histogram of oriented gradients (HOG) features [18]. The linear SVM classifier presented in Section V-B is applied to the features extracted by all three methods. In this binary ATR and classification problem there are four possible outcomes obtained by the SVM binary classifier. If the outcome from a prediction is positive and the actual value is also positive, then it is called a true positive (TP); however if the actual value is negative then it

TABLE I
ALEXNET ARCHITECTURE

Layer No.	Layer type	Description	Layer No.	Layer type	Description
1	'input'	Image Input	5	'conv4'	Convolution
2	'conv1'	Convolution		'relu4'	ReLU
	'relu1'	ReLU	6	'conv5'	Convolution
	'norm1'	Cross Channel Normalization		'relu5'	ReLU
	'pool1'	Max Pooling		'pool5'	Max Pooling
3	'conv2'	Convolution	7	'fc6'	Fully Connected
	'relu2'	ReLU		'relu6'	ReLU
	'norm2'	Cross Channel Normalization	8	'fc7'	Fully Connected
	'pool2'	Max Pooling		'relu7'	ReLU
4	'conv3'	Convolution	9	'fc8'	Fully Connected
			10	'prob'	Softmax
	'relu3'	ReLU	11	'classfication-Layer'	Classification Output

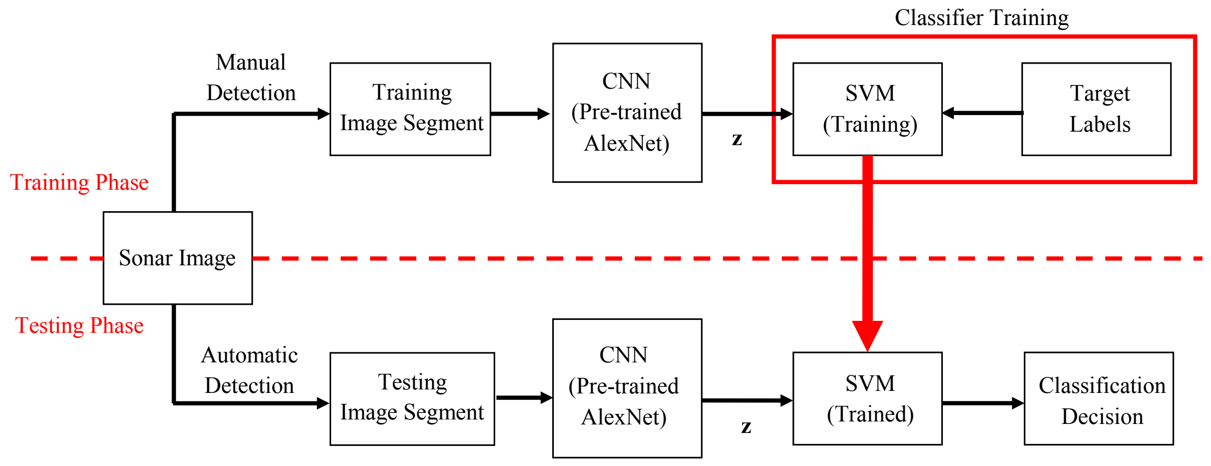


Fig. 12. Training architecture including training and testing phases.

TABLE II
SONAR-IMAGE GROUPS USED FOR CROSS VALIDATION

Group	$\mathcal{G}_1$	$\mathcal{G}_2$	$\mathcal{G}_3$	$\mathcal{G}_4$	$\mathcal{G}_5$
Sonar image index	1-8	9-13	14,16-19	15,20,21	28-35
No. of image segments	47	46	46	46	48

TABLE III
TOTAL CLASSIFICATION RESULTS

Methods	CNN+SVM	LBP+SVM	HOG+SVM
ACC	**0.9588**	0.9107	0.8351
TPR	**0.8696**	0.6812	0.5797

is said to be a false positive (FP). Conversely, a true negative (TN) has occurred when both the prediction outcome and the actual value are negative, and false negative (FN) is when the prediction outcome is negative while the actual value is positive. According to these concepts, the confusion matrix is defined as,

$$C = \begin{bmatrix} n_{TP} & n_{FN} \\ n_{FP} & n_{TN} \end{bmatrix} \tag{17}$$

where n_{TP}, n_{FN}, n_{FP}, and n_{TN} denote the number of the corresponding outcomes.

The classification accuracy (ACC) and the true positive rate (TPR) are defined to evaluate the performance the binary classification as follows

$$ACC = \frac{n_{TP} + n_{TN}}{n_{TP} + n_{FN} + n_{FP} + n_{TN}} \tag{18}$$

$$TPR = \frac{n_{TP}}{n_{TP} + n_{FN}}. \tag{19}$$

where ACC represents the general classification performance. For comparison, the ACC of all three methods is shown in Fig. 13, where the horizontal axis denotes the testing data sets, $\mathcal{G}_t$. The total performance, calculated based on all testing results, shows that the deep learning (CNN+SVM) method presented in this paper outperforms both LBP and HOG methods, by making better classification decisions across most testing data sets. Also, for comparison, the TPR of all three methods is shown in Fig. 14, where it can be seen that the proposed CNN+SVM method outperforms both LBP and HOG methods across most testing data sets. These results show that the features extracted by the AlexNet can describe the target objects in the sonar images better than the LBP and HOG features. It was also found that, unlike

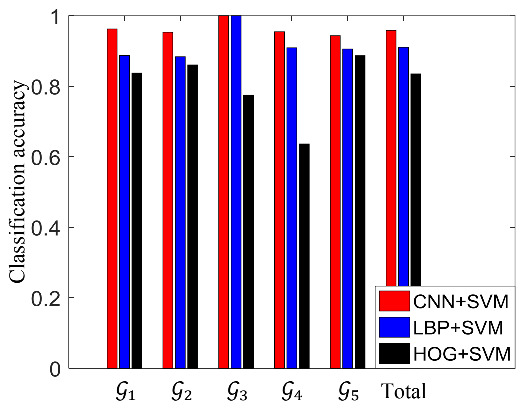


Fig. 13. Comparison of classification accuracies among different methods.

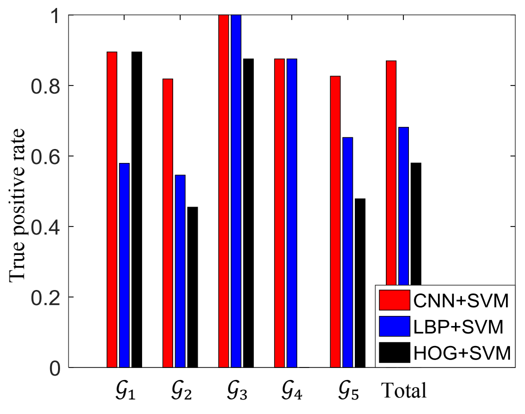


Fig. 14. Comparison of true positive rates among different methods.

CNN+SVM, the performance of HOG+SVM is not robust. For example, for the testing data set $\mathcal{G}_1$ the HOG+SVM achieves the same TPR as the proposed method, while for the testing data set $\mathcal{G}_4$ the HOG+SVM cannot find any target (TPR = 0). The performance comparison is also summarized in Table III, showing that the CNN+SVM is the best of the three approaches for ATR and classification.

VII. Conclusions and Future Directions

In this paper, it is demonstrated that by using deep learning feature extraction techniques, significant improvement in target recognition and classfication can be achieved for underwater sonar images, compared with using other feature extraction techniques such as histogram of oriented gradients (HOG) and local binary pattern (LBP). Sonar-driven path planning for autonomous UUVs and improving algorithm robustness for sonar images taken in different environment conditions are two possible directions of future research.

Acknowledgment

This work was supported by ONR grant N00014-15-1-2595. We thank Ziqi Yang for his effort in manually selecting and labeling the targets for classifier training.

References

[1] J. M. Bell, Y. R. Petillot, K. Lebart, S. Reed, E. Coiras, P. Y. Mignotte, and H. Rohou, "Target recognition in synthetic aperture and high resolution sidescan sonar," in *2006 IET Seminar on High Resolution Imaging and Target Classification*, Nov 2006, pp. 99–106.
[2] P. Blondel, *The Handbook of Sidescan Sonar*. Springer, 2009.
[3] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.
[4] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.
[5] Y. Bengio, "Learning deep architectures for ai," *Foundations and trends® in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.
[6] Y. Bengio, Y. LeCun *et al.*, "Scaling learning algorithms towards ai," *Large-scale kernel machines*, vol. 34, no. 5, 2007.
[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.
[8] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "Overfeat: Integrated recognition, localization and detection using convolutional networks," *arXiv preprint arXiv:1312.6229*, 2013.
[9] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, "Learning hierarchical features for scene labeling," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1915–1929, 2013.
[10] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.
[11] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1–9.
[12] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.
[13] R. Gonzalez and R. Woods, *Digital Image Processing*. Upper Saddle River, New Jersey: Prentice Hall, 2008.
[14] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net," in *workshop contribution at ICLR 2015*, 2015.
[15] K. P. Murphy, *Machine learning: a probabilistic perspective*. MIT press, 2012.
[16] Y. Anzai, *Pattern recognition and machine learning*. Elsevier, 2012.
[17] T. Ojala, M. Pietikainen, and D. Harwood, "Performance evaluation of texture measures with classification based on kullback discrimination of distributions," in *Pattern Recognition, 1994. Vol. 1-Conference A: Computer Vision & Image Processing., Proceedings of the 12th IAPR International Conference on*, vol. 1. IEEE, 1994, pp. 582–585.
[18] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1. IEEE, 2005, pp. 886–893.