Ship Detection in Spaceborne Optical Image with SVD Networks

Zhengxia Zou and Zhenwei Shi*, *Member IEEE*

Abstract

Automatic ship detection on spaceborne optical images is a challenging task, which has attracted wide attention due to its extensively potential applications in maritime security and traffic control. Although some optical image ship detection methods have been proposed in recent years, there are still three obstacles in this task, 1) the inference of clouds and strong waves, 2) difficulties in detecting both in-shore and off-shore ships, 3) high computational expenses. In this paper, we propose a novel ship detection method called SVD Networks (SVDNet), which is fast, robust and structurally compact. SVDNet is designed based on the recent popular Convolutional Neural Networks and the Singular Value Decompensation (SVD) algorithm. It provides a simple but efficient way to adaptively learn features from the remote sensing images. We evaluate our method on some spaceborne optical images of GaoFen-1 and Venezuelan Remote Sensing Satellites. The experimental results demonstrate that our method achieves high detection robustness and a desirable time performance in response to all above three problems.

Index Terms

Ship detection, Optical spaceborne image, Convolutional neural networks, Singular value decompensation.

I. Introduction

Ship detection is one of the hottest issues in the remote sensing image processing field due to both of its military and civil applications such as maritime surveillance, ship rescue and fishery management. There have been many earlier researches on ship detection, most of which were aimed at detecting ships in Synthetic Aperture Radar (SAR) images [1]. In recent years, with the rapid development of optical spaceborn imaging technology, some researchers have paid more attentions to detecting ships with optical images due to their good properties of higher resolution and more detailed spatial contents compared to SAR images.

This work was supported by the National Natural Science Foundation of China under Grant 61273245, the Beijing Natural Science Foundation under Grant 4152031, the funding project of State Key Laboratory of Virtual Reality Technology and Systems, Beihang University under Grant VR-2014-ZZ-02, the Fundamental Research Funds for the Central Universities under Grant YWF-14-YHXY-028 and Grant YWF-15-YHXY-003. *(Corresponding author: Zhenwei Shi.)*

Zhengxia Zou (e-mail: zhengxiazou@buaa.edu.cn) and Zhenwei Shi (Corresponding Author, e-mail: shizhenwei@buaa.edu.cn) are with Image Processing Center, School of Astronautics, Beihang University, Beijing 100191, China, and with Beijing Key Laboratory of Digital Media, Beihang University, Beijing 100191, China, and also with State Key Laboratory of Virtual Reality Technology and Systems, School of Astronautics, Beihang University, Beijing 100191, China.

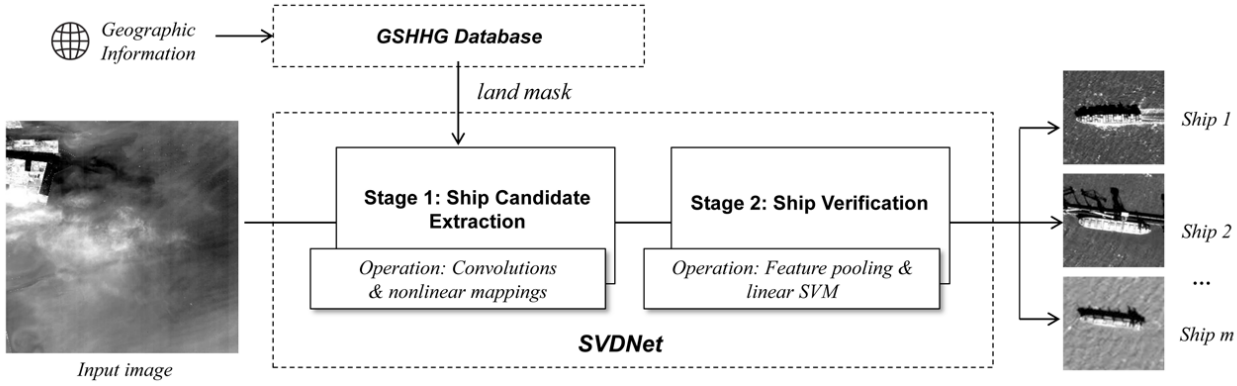


Fig. 1. Ship detection procedure of the proposed method.

In early times, due to the limited resolution of optical images, some researchers used ship track detection methods to identify ships [2, 3]. In recent years, some ship detection methods with high resolution optical spaceborn images have been proposed [4–20]. Most of these methods take a coarse-to-fine detection strategy, and their algorithm flow can be divided into the following two stages: 1) ship candidate extraction stage, 2) ship verification stage.

In the ship candidate extraction stage, the detection algorithm aims to search out all the candidate regions that possibly contain the ship targets. In this stage, some frequently used methods include the image segmentation based methods [4–12], the saliency detection based methods [9, 13, 14], the wavelet transformation based methods [6, 15] and the anomaly detection based method [16]. A common property of these methods is they are all designed in some unsupervised manners. Although most of them can give a nice candidate detection results in clear and clam sea environments, there are at least two situations these methods can not get satisfactory results. The first situation is detecting ships under clouds, and the second one is detecting ships near the port. Since there are no shape and structure priors of ships integrated in these methods, they may either cut a whole ship candidate into some small pieces or make the ship candidates merged to the nearby land or cloud regions. After the ship candidate extraction stage, there are usually some other postprocessing steps to exclude the backgrounds by morphological operation methods [12, 15] or neighborhood analysis methods [8]. But the parameters or the threshold values of these postprocessing methods are usually chosen by hand, which lacks the flexibility and the essential physical meaning.

In the ship verification stage, each individual ship candidate region is further verified whether it contains a real ship target. Previous works of this stage [4, 8, 11–13, 15–18] mainly take the advantage of some machine learning methods by converting the process of this stage into the feature extraction and the binary classification operations (targets VS backgrounds). The commonly used methods of this stage include support vector machine (SVM) [4, 8, 12, 13, 17], extreme learning machine [15], Adaboost [16], sparse representation [18] and the neural networks [11]. Apart from these supervised learning methods, some unsupervised methods are also used in this stage, such as the contour analysis method [6] and shape analysis method [9, 19].

The success of machine learning algorithms generally depends on a right way of data representation [21]. In the past two decades, some famous hand-craft features are proposed such as Gabor features [22], Local Binary Patterns (LBP) [23], Scale Invariant Feature Transform (SIFT) [24] and Histogram of Oriented Gradients (HOG) [25]. Some of the previous ship detection methods [8, 9, 12, 16] are also built based on these feature descriptors. Although these hand-crafted features have made great success for certain data and applications such as pedestrian detection and face recognition, designing effective features for new applications such as ship detection usually requires large amount of the domain knowledge, and these hand-crafted features cannot be simply adapted to these new conditions [26]. In recent years, automatically learning feature from data has become a popular way to go beyond the limitations of hand-crafted features. An example of such methods in image recognition field is Convolutional Neural Networks (CNN) [27], where multiple trainable convolutional layers and the nonlinear mapping layers are connected in series with one and another, and finally followed by a supervised classifier, such as SVM or softmax classifier. To learn the convolutional filters, some simple but efficient methods are proposed recently, such as wavelet scattering representation method [28] and Principle Component Analysis (PCA) representation method [26]. These methods use unsupervised learning techniques to represent source domain knowledge. In this paper, we propose a novel ship detection methods named SVD Networks (SVDNet), which is robust, structurally compact and theoretically sound. SVDNet is designed by the inspiration of the recent popular CNN structure and the Singular Value Decompensation (SVD) algorithm, where the SVD algorithm is used to automatically learn convolutional filters from a large amount of source domain training data. Since the features of our method are adaptively learned from ship and background image data, our method shows higher robustness and discriminative power.

In our work, we use totally different ship detection ideas compared to the previous methods. SVDNet integrates the prior knowledge of the ships and backgrounds by training large amount of images. Instead of using hand-crafted features, the features we used in our method are adaptively learned from spaceborn optical image data. The whole detection process of the proposed method is shown in Fig. 1. In the ship candidate detection stage, we use three convolutional layers and three nonlinear mapping layers to highlight all candidates of ship target and suppress the undesired backgrounds. Then, in the ship verification stage, each ship candidate is further to be verified whether it covers a real ship target by using feature pooling operation and the linear SVM classifier. In our method, we use the land mask to remove the land regions. The land mask of the input image is generated by the Global Self-consistent Hierarchical High-resolution Geography (GSHHG) database with the geographic information of the image. GSHHG database is created and published by the National Oceanic and Atmospheric Administration (NOAA) Geosciences Lab, and can be free-download from the Internet [30]. The GSHHG geography data includes coastlines, major rivers, and lakes of the world, and it comes in five different resolutions: crude, low, intermediate, high, and full. We use the full resolution data to generate the land mask.

The rest of this paper is organized as follows. In Section II, we will introduce the structure of our SVDNet. In section III, we will further give a detailed introduction of our feature learning algorithms in SVDNet. Some experimental results and analysis are given in section IV and section V. The conclusions are drawn in Section VI.

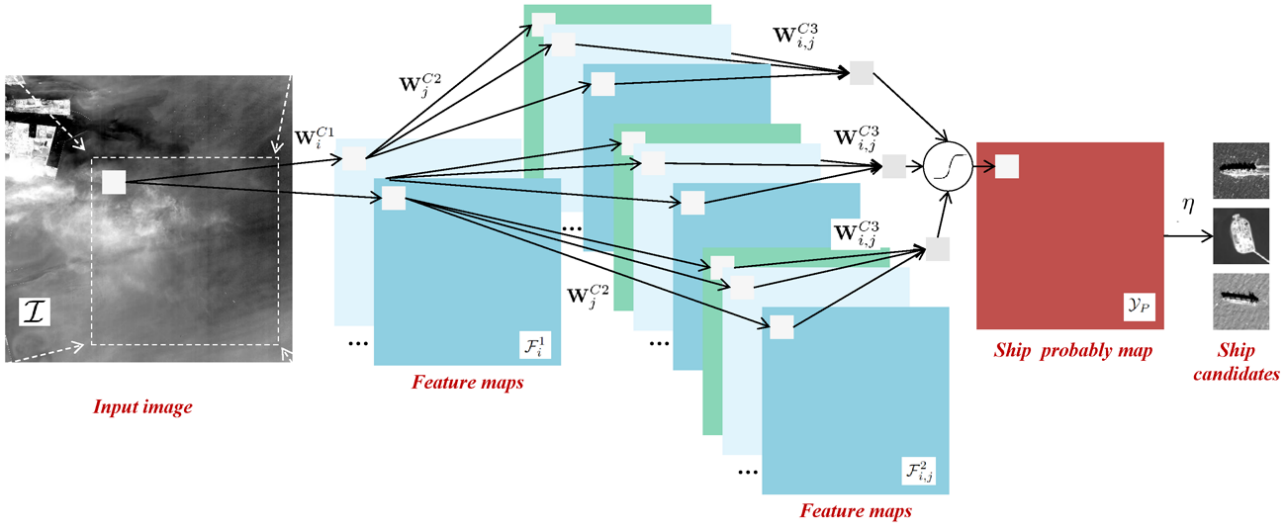


Fig. 2. A detailed ship detection procedure of SVDNet: Stage 1.

II. New Ship detection method: SVDNet

In this section, we will give a detailed introduction to the structure of SVDNet and explain how it works in the ship detection task.

A. Motivations

Most of the previous ship detection methods may face one of the following problems. Firstly, most of them use unsupervised methods to detect ship candidates, which may become unstable in complex environments. Some of these methods are empirically designed and have a large number of parameters. Some others consist of many trivial processing steps, where if one step fails to detect the ship candidates, the targets will not be recovered in the following processes. Secondly, most of them only focus on detecting off-shore ships, but both in-shore ships and off-shore ships are important for detection tasks[1]. Although there are some methods specifically designed for detecting in-shore ships [17, 20], few of them can deal with both of them very well. Thirdly, with the fast growing size of the remote sensing images, the time efficiency of the ship detection algorithm is become more and more important, which also brings new challenges to the ship detection tasks. Considering all above factors, we believe a good ship detection method should meet the following three requirements:

1) *Robustness*: The detection method should be robust to inference of illumination variations, clouds and waves.
2) *Generality*: The detection method should be able to detect both in-shore and off-shore ships.
3) *Computational efficiency*: The detection method should have low computational expenses. This is required especially for large scale remote sensing images.

[1]In this paper, the in-shore ships refers to the ships near the land or adjacent with the port. The off-shore ships refers to the ships far away from the land.

Unfortunately, to our best knowledge, no previous ship detection methods can satisfy all these goals at the same time. On comprehensive consideration of above factors, we propose SVDNet for ship detection. In the next two subsections, we will give a detailed introduction of SVDNet.

B. Stage 1: Ship Candidate Extraction

We use a multiscale convolutional structure to obtain the ship candidates. The structure of in this stage is shown in Fig. 2. Suppose that we are given an test images $\mathcal{I}$ with the size of $H \times W$, and the filter banks of the 1^{st}, 2^{nd} and 3^{rd} convolutional layers are represented as

$$\begin{aligned} &\mathbf{W}_i^{C1} \in \mathbb{R}^{k_1 \times k_1}, i = 1, 2, \ldots, L_1 \\ &\mathbf{W}_i^{C2} \in \mathbb{R}^{k_2 \times k_2}, i = 1, 2, \ldots, L_2 \\ &\mathbf{W}_i^{C3} \in \mathbb{R}^{k_3 \times k_3}, i = 1, 2, \ldots, L_1 \times L_2, \end{aligned} \tag{1}$$

respectively, where L_1, L_2 and $L_1 \times L_2$ represent the number of filters in the 1^{st}, 2^{nd} and 3^{rd} convolutional layers, and k_1, k_2 and k_3 represent their convolutional filter sizes. In Fig. 2, $C1$, $C2$ and $C3$ represent the 1^{st}, 2^{nd} and 3^{rd} convolutional layers, and $M1$, $M2$ and $M3$ represent the 1^{st}, 2^{nd} and 3^{rd} nonlinear mapping layers. Suppose $\mathcal{F}_i^1$ and $\mathcal{F}_i^2$ represent the ith output map of the 1^{st} and 2^{nd} nonlinear mapping layers, and $\mathcal{Y}_P$ represents the output map of the 3^{rd} nonlinear mapping layer. Then we have

$$\begin{aligned} &\mathcal{F}_i^1 = f(\mathcal{I} \otimes \mathbf{W}_i^{C1}),\ i = 1, \cdot\cdot, L_1, \\ &\mathcal{F}_{i,j}^2 = f(\mathcal{F}_i^1 \otimes \mathbf{W}_j^{C2}),\ i = 1, \cdot\cdot, L_1, j = 1, \cdot\cdot, L_2, \\ &\mathcal{Y}_P = f(\sum_{i,j} \mathcal{F}_{i,j}^2 \otimes \mathbf{W}_{i,j}^{C3}),\ i = 1, \cdot\cdot, L_1, j = 1, \cdot\cdot, L_2. \end{aligned} \tag{2}$$

where $f(\cdot)$ represents the element-wise nonlinear mapping function

$$f(\theta) = 1/(1 + e^{-\theta}), \tag{3}$$

and $\otimes$ denotes the 2D convolution operation. In this paper, we call the filters $\mathbf{W}_i^{C1}$, $\mathbf{W}_j^{C2}$ as *primary filters* and we call the filters $\mathbf{W}_{i,j}^{C3}$ as the *candidate target filters*. We call the output maps $\mathcal{F}_i^1$ and $\mathcal{F}_j^2$ as the *feature maps*, and we call the output map $\mathcal{Y}_P$ as the *ship probability map* (see table I for more details).

In the first convolutional layer C1, we have L_1 primary filters, thus we get L_1 feature maps. While, in the second convolutional layer, since each feature map is regarded as the an individual input, thus this time we have L_1 groups of feature maps, that is to say, we have totally $L_1 \times L_2$ feature maps. The third convolutional layer C3 takes the $L_1 \times L_2$ feature maps of the second layer as its inputs, and it gives $L_1 \times L_2$ score maps as its outputs. After that, all score maps are added and then transformed by the non-linear mapping transformation. Finally, the each pixel value of the input image is mapped into the range of $(0, 1)$, which can be seen as the probability of how likely the convolutional window covers a ship target. By using this multilayer convolutional structure, we can obtain hierarchical feature responses on these filters by the multilayer convolutional process. In SVDNet, the C1, M1, C2 and M2 can be seen as the primary feature extraction layers, and C3 and M3 can be seen as the discriminative

layers. In this way, we obtain a ship probability map $\mathcal{Y}_P$ where all the potential ship targets are highlighted and the undesired backgrounds are suppressed.

After that, the ship probability map $\mathcal{Y}_P$ is compared with a threshold η, if the probability score $\mathcal{Y}_P(x, y)$ of the pixel $p(x, y)$ is greater than η, the current convolutional window is identified as a candidate target window, and need to be further processed, else, the current window is identified as a background. To detect the ships of different scales, we down-sample the original image to different scales while keep the filter banks as the fixed sizes. To further reduce the false alarm numbers, the non-maximal suppression [31] is used to select a small set of ship candidates from ship probability map $\mathcal{Y}_P$. Finally, the ship candidate windows of different scales are merged to form the final ship candidate detection results.

TABLE I
THE DEPLOYMENT OF SVDNET

Structure unit	Name	Formulation	Size	Number
$\mathcal{I}$	*input image*	-	$H \times W$	-
$\mathbf{W}_i^{C1}$	*primary filters*	see Algorithm 1	$k_1 \times k_1$	$i = 1, \ldots, L_1$
C1	*convolutional layer 1*	$\mathcal{I} \otimes \mathbf{W}_i^{C1}$	-	-
M1	*nonlinear mapping layer 1*	$f(\theta) = 1/(1 + e^{-\theta})$	-	-
$\mathcal{F}_i^1$	*feature maps*	$\mathcal{F}_i^1 = f(\mathcal{I} \otimes \mathbf{W}_i^{C1})$	$H \times W$	$i = 1, \ldots, L_1$
$\mathbf{W}_j^{C2}$	*primary filters*	see Algorithm 1	$k_2 \times k_2$	$j = 1, \ldots, L_2$
C2	*convolutional layer 2*	$\mathcal{F}_i^1 \otimes \mathbf{W}_j^{C2}$	-	-
M2	*nonlinear mapping layer 2*	$f(\theta) = 1/(1 + e^{-\theta})$	-	-
$\mathcal{F}_{i,j}^2$	*feature maps*	$\mathcal{F}_{i,j}^2 = f(\mathcal{F}_i^1 \otimes \mathbf{W}_j^{C2})$	$H \times W$	$i = 1, \ldots, L_1,\ j = 1, \ldots, L_2$
$\mathbf{W}_{i,j}^{C3}$	*candidate target filters*	see Algorithm 2	$k_3 \times k_3$	$i = 1, \ldots, L_1,\ j = 1, \ldots, L_2$
C3	*convolutional layer 3*	$\mathcal{F}_{i,j}^2 \otimes \mathbf{W}_{i,j}^{C3}$	-	-
M3	*nonlinear mapping layer 1*	$f(\theta) = 1/(1 + e^{-\theta})$	-	-
$\mathcal{Y}_P$	*ship probability map*	$f(\sum_{i,j} \mathcal{F}_{i,j}^2 \otimes \mathbf{W}_{i,j}^{C3})$	$H \times W$	1
$\mathcal{D}_i$	*hash maps*	$\mathcal{D}_i = \sum_{j=1}^{L_2} 2^{j-1} H(\mathcal{F}_{i,j}^2)$	$H \times W$	$i = 1, \ldots, L_1$

C. Stage 2: Ship Verification

After getting a set of ship candidate windows, each window is analyzed by more discriminative features to verify whether it really contains a ship instance. The structure of the proposed method in this stage is shown in Fig. 3 (see table I for more details).

In this stage, we use feature maps $\mathcal{F}_{i,j}^2$, the output of M2 layer, as the basic features of each ship candidate. We first crop the feature maps based on the location and the size of each ship candidate. Then, the feature maps of each ship candidate is resized to the $S \times S$ images to be further processed. In this case, the ship verification task can be seen as a feature extraction and a basic binary-class classification problem, where each sample image can be treated as an independent instance to be classified into the "target class" and the "background class". To obtain better illumination invariance ability, we first binarize the feature maps by $H(\mathcal{F}_{i,j}^2)$, where $H(\cdot)$ is an element-wise

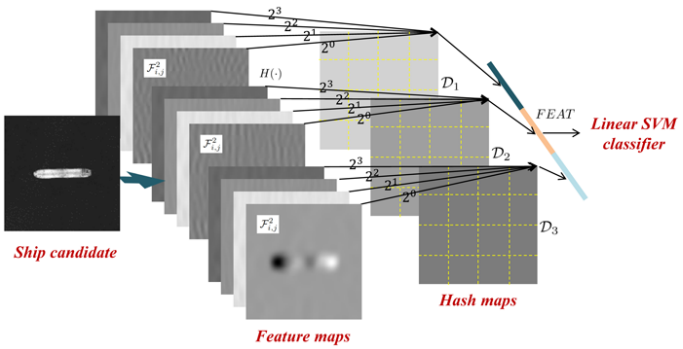


Fig. 3. A detailed ship detection procedure of SVDNet: Stage 2. In this figure we use $L_1 = 3$, $L_2 = 4$ and $S = 4B$ for an example.

binarize function $H(t) = \{^{1,\ t\geq 0.5}_{0,\ t<0.5}$. Then, each pixel of the feature map can be seen as a binary vector with $L_1 \times L_2$ elements. For the ith group of the elements in the pixel vector, we encode the binary vector into a decimal number by

$$\mathcal{D}_i = \sum_{j=1}^{L_2} 2^{j-1} H(\mathcal{F}_{i,j}^2),\ \ i = 1, 2, \ldots, L_1, \tag{4}$$

which converts the feature maps $\mathcal{F}_{i,j}^2$ back into several single integer-valued "images" $\mathcal{D}_i$. Since this can be simply viewed as a binary hash coding process, we called $\mathcal{D}_i$ as *hash maps*, where the value of each pixel is in the range of $[0, 2^{L_2} - 1]$. Next, to obtain better shift invariance ability, the hash maps are evenly divided into different blocks with the block size of $B \times B$ pixels. Then, the histogram of a hash map in each block is accumulated, and histograms of all blocks are concatenated into one feature vector $Hist(\mathcal{D}_i) \in \mathbb{R}^{2^{L_2-1}S^2/B^2 \times 1}$. At last, feature vectors of all hash maps are further concatenated together to form the final feature descriptor of the image sample

$$FEAT = [Hist(\mathcal{D}_1); Hist(\mathcal{D}_2); \ldots; Hist(\mathcal{D}_{L_1})]. \tag{5}$$

The dimension of the final feature descriptor is

$$dim(FEAT) = L_1 2^{L_2-1} S^2 / B^2. \tag{6}$$

On ship verification stage, accuracy and time efficiency are both important factors. The Support Vector Machine (SVM) [32] is a popular algorithm for classification problems. SVM is a kind of supervised learning method which draw optimal hyperplane with the largest margin to separate the data into two classes. It has many good properties such as good generalization ability and rich geometric interpretations. The feature descriptor produced by SVDNet provides a high discriminative power, thus, considering the computational efficiency, we choose linear SVM as the classifier in the very end of the ship detection process. In this paper, we use LibLinear package [33] for the SVM training process.

It should be noticed that although some design ideas and structure elements of our method are borrowed from PCANet [26] (e.g. the block-wise histograms and hash coding), there are essential differences between them: 1) Their target problems are different. The PCANet is a kind of feature extraction method which is initially designed for

classification problem. However, the proposed SVDNet is a kind of target detection method, which contains a series of detection steps including candidate target detection, target verification and etc. 2) Their network structures are different. The PCANet consists of two mean-removed layers and two convolution layers, while SVDNet consists of three convolution layers and three nonlinear mapping layers. In addition, to detect ships of different scales, SVDNet also introduces the multi-scale down sampling operation. 3) Their input and output are different. The input of PCANet is a sample image, and its output is the corresponding feature vector, while the input of SVDNet is the whole remote sensing image and its output is the bounding boxes of all ship targets. 4) Their learning algorithms are different. The PCANet uses unsupervised feature learning algorithm to learn convolutional filters, while the SVDNet uses unsupervised + supervised feature learning algorithms. The learning algorithm will be introduced in the next section.

III. Feature Learning

In SVDNet, the filter banks $\mathbf{W}_i^{C1}$, $\mathbf{W}_i^{C2}$ and $\mathbf{W}_i^{C3}$ need to be adaptively learned from the spaceborn optical image data. In this section, we first introduce the unsupervised feature learning algorithm to learn the primary filters of C1 and C2 layers. Then we will introduce the supervised feature learning algorithm to learn the discriminative filters of C3 layer.

A. Learning Primary Filters in C1 and C2 Layers

Feature learning is a way to take advantage of data prior knowledge to compensate for the weakness of the hand-crafted features. In SVDNet, we use a simple but efficient unsupervised algorithm, SVD, to represent the data knowledge in C1 and C2 layers. Compared to other feature learning methods such as PCA [26] and wavelet scattering representation method [28], SVD is more generally applicable and have more extensive physical meanings [29]. In linear algebra, SVD is a factorization of a matrix with arbitrary size into three matrices. Suppose we have randomly selected n overlapped patches $\mathcal{P}_i^{C1}$ of size $k_1 \times k_1$ from m original images, $i = 1, 2, \ldots, n$. The pixels of each image patch $\mathcal{P}_i^{C1}$ can be reshaped into a patch vector $\mathbf{p}_i^{C1} \in \mathbb{R}^{k_1^2 \times 1}$, and all patches can be are arranged into a data matrix $\mathbf{P}_{C1} = [\mathbf{p}_1^{C1}, \mathbf{p}_2^{C1}, \ldots, \mathbf{p}_n^{C1}] \in \mathbb{R}^{k_1^2 \times n}$. Formally, the SVD of the data matrix $\mathbf{P}_{C1}$ is a factorization of the form

$$\mathbf{P}_{C1} = \mathbf{U}_{C1}\mathbf{\Sigma}_{C1}\mathbf{V}_{C1}^T, \tag{7}$$

where $\mathbf{U}_{C1} \in \mathbb{R}^{k_1^2 \times k_1^2}$ and $\mathbf{V}_{C1} \in \mathbb{R}^{n \times n}$ are two orthogonal matrices, $\mathbf{\Sigma}_{C1} \in \mathbb{R}^{k_1^2 \times n}$ is a rectangular diagonal matrix with non-negative real numbers on its diagonal. The diagonal elements of $\mathbf{\Sigma}_{C1}$ are known as the singular values of $\mathbf{P}^{C1}$. The columns of $\mathbf{U}_{C1}$ and the columns of $\mathbf{V}_{C1}$ are called the left-singular vectors and right-singular vectors of $\mathbf{P}_{C1}$, respectively. In this paper, the left-singular vectors $\mathbf{u}_i^{C1}$ of $\mathbf{U}_{C1}$ with the largest L_1 singular values are chosen as the primary filter banks of C1 layer. The filter banks $\mathbf{W}_i^{C1}$ can be finally obtained by reforming $\hat{\mathbf{u}}_i^{C1}$ back into the 2D filters as follows

$$\mathbf{u}_i^{C1} \xrightarrow{\text{reshape}} \mathbf{W}_i^{C1}, i = 1, \cdot\cdot, L_1. \tag{8}$$

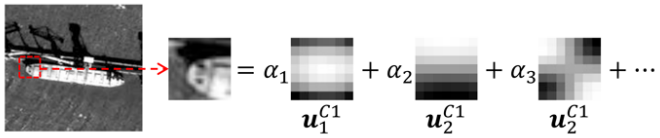


Fig. 4. By using SVD algorithm, we can adaptively obtain a set of orthogonal bases where each small patch of an input image can be decomposed as the linear combination of these bases. In SVDNet, we use these these bases with the largest singular value as the convolutional filters.

Similarly, in C2 layer, we randomly select n image patches $\mathcal{P}_i^{C2}$ from the feature maps $\mathcal{F}_i^1$ after the C1 and M1 operations, and we have the data matrix $\mathbf{P}_{C2}$ this time, and we have

$$\mathbf{P}_{C2} = \mathbf{U}_{C2}\mathbf{\Sigma}_{C2}\mathbf{V}_{C2}^T. \tag{9}$$

The column vectors $\mathbf{u}_i^{C2}$ with the largest L_2 singular values are used as the filter banks of C2 layer,

$$\mathbf{u}_i^{C2} \xrightarrow{\text{reshape}} \mathbf{W}_i^{C2}, i = 1, \cdot\cdot, L_2. \tag{10}$$

Algorithm 1 gives the feature learning algorithm in C1 and C2 layers.

Algorithm 1 SVD algorithm for C1 and C2 layers

Initialization: Set filter number L_1, and L_2.

Input: m original images $\mathcal{I}_1, \ldots, \mathcal{I}_m$.

1. Learning filters of C1

1.1. Randomly select n $k_1 \times k_1$ patches from $\mathcal{I}_1 \sim \mathcal{I}_m$:

$\mathbf{P}_{C1} = [\mathbf{p}_1^{C1}, \mathbf{p}_2^{C1}, \ldots, \mathbf{p}_n^{C1}] \in \mathbb{R}^{k_1^2 \times n}$.

1.2. Compute SVD: $\mathbf{P}_{C1} = \mathbf{U}_{C1}\mathbf{\Sigma}_{C1}\mathbf{V}_{C1}^T$.

2. Learning filters of C2

2.1. Convolution and nonlinear transformation:

$\mathcal{F}_i^1 = f(\mathcal{I} \otimes \mathbf{W}_i^{C1}),\ i = 1, \cdot\cdot, L_1$

2.2. Randomly select n $k_2 \times k_2$ patches from $\mathcal{F}_i^1$:

$\mathbf{P}_{C2} = [\mathbf{p}_1^{C2}, \mathbf{p}_2^{C2}, \ldots, \mathbf{p}_n^{C2}] \in \mathbb{R}^{k_2^2 \times n}$.

2.3. Compute SVD: $\mathbf{P}_{C2} = \mathbf{U}_{C2}\mathbf{\Sigma}_{C2}\mathbf{V}_{C2}^T$.

Output: Basis vectors $\mathbf{u}_1^{C1}, \ldots, \mathbf{u}_{L_1}^{C1}$ and $\mathbf{u}_1^{C2}, \ldots, \mathbf{u}_{L_2}^{C2}$.

Remark 1: By using SVD, we obtain a set of orthogonal bases of the image data, which describes the basic 2D structural elements of ship targets and backgrounds. Since $\mathbf{U}_{C1}$ and $\mathbf{U}_{C2}$ are two orthogonal matrices, every image patch $\mathbf{p}^{Cl}$ can be decomposed as the linear combination of these orthogonal bases,

$$\mathbf{p}^{Cl} = \alpha_1\mathbf{u}_1^{Cl} + \alpha_2\mathbf{u}_2^{Cl} + \cdots + \alpha_{k_l^2}\mathbf{u}_{k_l^2}^{Cl},\ l = 1, 2. \tag{11}$$

which is shown in Fig. 4. Thus the inner product of an input image patch and one of these bases turns out to be the corresponding linear coefficient α_i in equation (11)

$$\begin{aligned}\langle \mathbf{W}_i^{Cl}, \mathcal{P}^{Cl} \rangle &= \langle \mathbf{u}_i^{Cl}, \mathbf{p}^{Cl} \rangle \\ &= \langle \mathbf{u}_i^{Cl}, (\alpha_1 \mathbf{u}_1^{Cl} + \cdots + \alpha_{k_l^2} \mathbf{u}_{k_l^2}^{Cl}) \rangle \\ &= \alpha_i, \ l = 1, 2.\end{aligned} \tag{12}$$

Remark 2: Actually, the 2D convolution of an input image with a convolutional filter is almost the same as the inner product of the filter with all the densely overlapped image patches of this image. The only difference is we need to rotating the filter 180° before convolution. In this way, the 2D convolutional responses of an input image with one of these filters turn out to be the coefficient maps of these orthogonal bases. That is to say, by taking these orthogonal bases as our filter banks, the convolutional process provides an adaptive and effective hierarchical feature representation of the input image.

B. Learning Discriminative Filters in C3 Layer

The aim of the C3 layer is to highlight the ship candidates and suppress the undesired backgrounds. Since we have to teach our network what a ship target looks like, we use supervised learning algorithm in the third layer to obtain the filter banks $\mathbf{W}_j^{C3}$. To better understand this, the kernels of the third layer can be simply considered as the ship target templates, and the convolutions of this layer can be considered as the template matching operations. The convolution operations of C3 layer can be equally simplified by the following vector inner product expression

$$y = \mathbf{w}^T \mathbf{x}^{C3}, \tag{13}$$

where $\mathbf{w} = [\mathbf{w}_{1,1}^{C3}; \ldots; \mathbf{w}_{L_1,L_2}^{C3}] \in \mathbb{R}^{L_1 L_1 k_3^2 \times 1}$ represents the filter banks of C3 layer, and $\mathbf{x}^{C3} = [\mathbf{x}_{1,1}^{C3}; \ldots; \mathbf{x}_{L_1,L_2}^{C3}] \in \mathbb{R}^{L_1 L_1 k_3^2 \times 1}$ represents the detection window of the feature maps. After the nonlinear transformation of the M3 layer, the score $f(y)$ is mapped to (0,1) and can be considered as the possibility that the detection window $\mathbf{x}^{C3}$ covers a ship target

$$\begin{aligned}P(+1|\mathbf{x}^{C3}, \mathbf{w}) &= f(\mathbf{w}^T \mathbf{x}^{C3}) \\ &= 1/(1 + e^{-\mathbf{w}^T \mathbf{x}^{C3}}).\end{aligned} \tag{14}$$

Similarly, the possibility that the detection window $\mathbf{x}^{C3}$ do not cover a ship target can be written as

$$\begin{aligned}P(-1|\mathbf{x}^{C3}, \mathbf{w}) &= 1 - P(+1|\mathbf{x}^{C3}, \mathbf{w}) \\ &= 1/(1 + e^{\mathbf{w}^T \mathbf{x}^{C3}}).\end{aligned} \tag{15}$$

Suppose we have collected N_t target windows and N_b background windows, then the log-likelihood of all windows can be represented as

$$\begin{aligned} log(L(\mathbf{w})) &= log(\prod_{i=1}^{N} P(y_i|\mathbf{x}_i^{C3}, \mathbf{w})) \\ &= log(\prod_{i=1}^{N} 1/(1+e^{-y_i\mathbf{w}^T\mathbf{x}_i^{C3}})) \\ &= -\sum_{i=1}^{N} log(1+e^{-y_i\mathbf{w}^T\mathbf{x}_i^{C3}}), \end{aligned} \tag{16}$$

where $N = N_t + N_b$ is the total number of windows. $y_i = \{+1, -1\}$ represents the label of the target and background windows. The filters of C3 layer can be easily learned by the Maximum Likelihood Estimation (MLE). Since maximizing the log-likelihood $log(L(\mathbf{w}))$ is just equivalent to minimizing the negative log-likelihood $-log(L(\mathbf{w}))$, the feature learning process of this layer can be finally represented by solving the following unconstrained optimization problem

$$\min_{\mathbf{w}} J(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N} log(1+e^{-y_i\mathbf{w}^T\mathbf{x}_i^{C3}}) + \beta\|\mathbf{w}\|_2^2, \tag{17}$$

where β is a positive weight parameter. The objective function $J(\mathbf{w})$ consists of two parts, where the first part $\sum_{i=1}^{N} log(1+exp(-y_i\mathbf{w}^T\mathbf{x}_i^{C3}))$ represents the negative log-likelihood, and the second part $\beta\|\mathbf{w}\|_2^2$ is a regularization term to reduce the model's complexity and also to increase its generalization ability. We collect a large number of candidate windows including in-shore and off-shore ships of different shapes, appearances and in different illuminations. In this case, the problem (17) can be efficiently solved by the Stochastic Gradient Descent (SGD) method [29]. SGD and its variants are most commonly used algorithm for large scale optimization problems. It is very similar to conventional (batch) gradient descent method, except that the gradient is randomly estimated to perform its update. Concretely, the gradient of (17) with a single input $\mathbf{x}_i^{C3}$ can be represented as follows

$$\nabla_{\mathbf{w}} J(\mathbf{w}; \mathbf{x}_i^{C3}) = -\frac{y_i e^{-y_i\mathbf{w}^T\mathbf{x}_i^{C3}}}{1+e^{-y_i\mathbf{w}^T\mathbf{x}_i^{C3}}}\mathbf{w} + 2\beta\mathbf{w}. \tag{18}$$

In SVDNet, we use the mini-batch SGD algorithm to solve this problem, where in each iteration, we use a small subset of the samples (mini-batch) instead of single sample to estimate the gradient. After we obtain the optimal solution $\hat{\mathbf{w}} = [\hat{\mathbf{w}}_{1,2}^{C3}; \ldots; \hat{\mathbf{w}}_{L_1,L_2}^{C3}]$, the filter banks $\mathbf{W}_{i,j}^{C3}$ can be finally obtained by reforming $\hat{\mathbf{w}}^{C3}$ back into L_1 groups of 2D filters as follows

$$\begin{gathered} \hat{\mathbf{w}}_{i,j}^{C3} \xrightarrow{\text{reshape}} \mathbf{W}_{i,j}^{C3}, \\ i = 1, \ldots, L_1,\ j = 1, \ldots, L_2. \end{gathered} \tag{19}$$

Algorithm 2 gives the feature learning algorithm in C3 layer.

Algorithm 2 mini-batch SGD algorithm for C3 layer

Initialization: Initialize $\mathbf{w}$. Set learning rate $\mu > 0$, maximum iteration number $T > 0$ and min-batch number $M > 0$.

Input: Training data $\mathbf{x}_i$ and labels $y_i = \{+1, -1\}$.

for $t = 1, 2, \ldots, T$ **do**

1. Randomly choose a minibatch set of M samples
   $\mathrm{S} = \{\mathbf{x}_1^{C3}, \mathbf{x}_2^{C3}, \ldots, \mathbf{x}_M^{C3}\}$.
2. Compute the gradient of M samples:
   $\nabla_{\mathbf{w}} J(\mathbf{w}; \mathrm{S}) = -\frac{1}{M} \sum_{\mathrm{S}} y_i \frac{exp(-y_i \mathbf{w}^T \mathbf{x}_i^{C3})}{1+exp(-y_i \mathbf{w}^T \mathbf{x}_i^{C3})} \mathbf{w} + \frac{2\beta}{M} \mathbf{w}$.
3. Update $\mathbf{w}$:
   $\mathbf{w} \leftarrow \mathbf{w} - \mu \nabla_{\mathbf{w}} J(\mathbf{w}; \mathrm{S})$

end for

Output: $\mathbf{w}$.

TABLE II

DETAILED INFORMATION OF THE EXPERIMENTAL IMAGES

ID	Satellite	Mode	Image Size (pixel)	XY-Resolution	Topleft Lat.	Topleft Lon.	Landscape	Usage
1	GF-1	PAN	18192×18000	2.5m×2m	39.0605°	118.177°	ocean and harbour	train
2	GF-1	PAN	18192×18000	2.5m×2m	39.2687°	118.103°	ocean and farmland	test
3	GF-1	PAN	18192×18000	2.5m×2m	38.9864°	118.545°	ocean and harbour	train
4	GF-1	PAN	18192×18000	2.5m×2m	39.2635°	118.628°	ocean and farmland	test
5	GF-1	PAN	18192×18164	2.5m×2m	38.9878°	117.804°	ocean and harbour	train
6	GF-1	PAN	18192×18164	2.5m×2m	39.5437°	119.105°	ocean and paddy field	test
7	VRSS-1	PAN	14154×14266	2.5m×2.5m	10.7384°	−68.0561°	ocean and city	train
8	VRSS-1	PAN	14132×14239	2.5m×2.5m	10.2769°	−64.9345°	ocean and city	test
9	VRSS-1	PAN	14364×14441	2.5m×2.5m	10.2793°	−65.0075°	ocean and city	train
10	VRSS-1	PAN	14508×14616	2.5m×2.5m	10.2815°	−64.9017°	ocean and city	test
11	VRSS-1	PAN	14133×14239	2.5m×2.5m	10.7385°	−67.2836°	ocean and hills	train
12	VRSS-1	PAN	14328×14412	2.5m×2.5m	10.9702°	−68.3355°	ocean and islands	test
13	VRSS-1	PAN	14078×14150	2.5m×2.5m	10.9669°	−64.1251°	ocean and islands	train
14	VRSS-1	PAN	13894×13967	2.5m×2.5m	10.2747°	−64.8712°	ocean and city	test
15	VRSS-1	PAN	13695×13807	2.5m×2.5m	10.7330°	−68.0578°	ocean and islands	train

IV. EXPERIMENTAL RESULTS

In this section, a set of spaceborne optical images of Venezuelan Remote Sensing Satellite (VRSS-1) and GaoFen-1 (GF-1) satellites are used to demonstrate the efficiency of SVDNet. Since there is no public ship detection dataset, and no one has published their source-code or software, we do not directly compare with other methods.

A. Experiment Setup

We evaluate our algorithm on six panchromatic images of GF-1 satellite and nine panchromatic images of VRSS-1 satellite. These images are taken during different time periods of a day with different light conditions. They contain the most types of the coastal landscapes such as the ocean, the city and the island. Among these images, eight of them (with odd image ID) are used for train, and the rest (with even image ID) are used for test. A detailed information of these images are listed in table II. Before ship detection, the binary land masks of these images are generated by using GSHHG database and geographical information of the input image. It should be noticed that since the raw pixel data of the original image is 16-bit depth, thus all images are converted to 8-bit ones by 2% linear stretch before the detection process. Apart from that, we do not apply any other pre-processing operations.

All ships targets in these images, including the in-shore ships and off-shore ships, have been labeled manually. In our experiments, some ambiguous ships are not labeled and are excluded from the target set. These ones refer to the ships whose length are smaller than 20 pixels or partially outside the image. Although some small ships may have very large ship wakes, we only label those ships with their body lengths larger 20 pixels. Some ships that clustered by clouds and can not be easily identified by human eyes are also excluded. These ones particularly refer to those ships whose clustered parts are larger than 2/3 of their sizes and cannot be easily identified unless by an experienced interpreter. Eliminating all these invalid ship instances, a total of 606 ships are labeled for experiments. Apart from these images, we also select 1,800 ship samples from other 102 image slices of Google Earth. These ships are used as extra training samples to provide additional information of the ship targets and enhance the classifier's generalization ability. Before training, the data augmentation is performed. For a positive sample image, we first crop four sub-images with 80% of its size at the left-top, right-top, left-bottom and right-bottom corners. Then, we resize these sub-images to their original sizes to enhance the classifiers shift-invariance and scale-invariance ability. Finally, including the original sample image, we obtain (606+1,800)×5=12,030 augmented positive samples in total. To detect ships in different directions, we also rotate each sample for 8 times (each time for 45 degrees), where an individual classifier is trained in each direction. The negative training samples are a set of ship-free randomly sampled patches from the GF-1, VRSS-1 and the Google Earth images. In this way, 12,030 positive samples and 17,000 negative samples are finally used for training the linear SVM classifier. To detect the ships of different sizes, each image is down-sampled at 14 scales from 1.0 to 10.0 with the unified down-sampled rate at the very beginning of the ship detection process. A detailed parameter setting of SVDNet is listed in table III. Since the filter size of C3 is set to $k_3 = 20$ pixels, the smallest ship size and largest ship size that can be detected by our method are $l_{min} = k_3 \times 1 = 20$ pixels and $l_{max} = k_3 \times 10 = 200$ pixels[2].

[2]Since the resolution of the test images are $2m/pxl$ and $2.5m/pxl$, the detection window can cover most types of the medium and large ships ($40m \sim 500m$). To detect smaller ships, we suggest using higher resolution images.

TABLE III
PARAMETER SETTING OF SVDNET

Stage	Parameter	Value
Stage 1	filter size $k_1 \times k_1$	7×7
	filter size $k_2 \times k_2$	7×7
	filter number L_1	8
	filter number L_2	4
	filter size $k_3 \times k_3$	20×20
	threshold value η	0.6
Stage 2	sample size $S \times S$	80×80
	block size $B \times B$	16×16

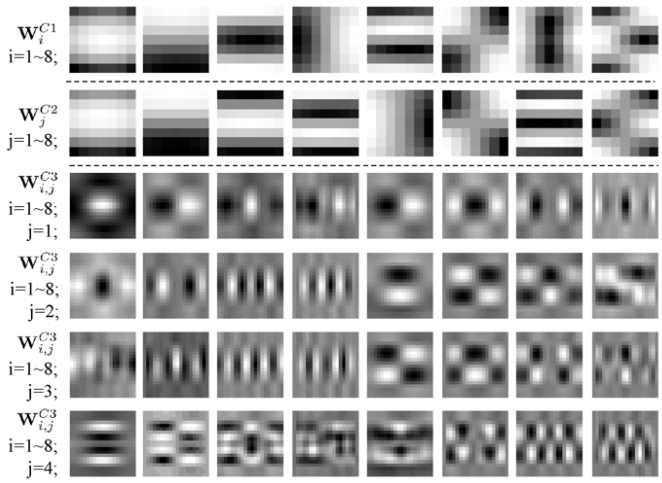


Fig. 5. Convolutional filters of C1, C2 and C3 that automatically learned by Algorithm 1 and Algorithm 2. The 1^{st} row shows the $1^{st} - 8^{th}$ filters of C1 layer. The 2^{nd} row shows the $1^{st} - 8^{th}$ filters of C2 layer. The $3^{rd} - 6^{th}$ rows show the $8 \times 4 = 32$ filters of the C3 layer.

B. Overall Results Statistics

In Algorithm 1, we use $n = 80,000$ image patches randomly selected from ship targets and backgrounds to learn the filters. The filter sizes are set to $k_1 = k_2 = 7$. In Algorithm 2, we set learning rate $\mu = 0.001$, regularization coefficient $\beta = 0.01$ and the maximum iteration number $T = 40,000$. We use all the positive and negative training samples to learn the filters of C3 layer. Fig. 7 shows the value of the objective function (17) of different iterations with different choices of mini-batch number $M = 10$, 20 and 100. In this figure, the horizontal axis values are set logarithmic scales for a clearer comparison. We can see that when we choose larger mini-batch number M, the objective function value will decrease faster and become smoother. However, setting larger values of batch-number M does not mean better. When M is larger than 20, we do not see any obvious improvements of the convergence speed, but the training time cost has increased quickly from $12.6s$ to $112.1s$. Thus in this paper, we set $M = 20$.

Fig. 5 shows the first eight filers of C1 and C2 layers that automatically learned by Algorithm 1, and the $8 \times 4 = 32$

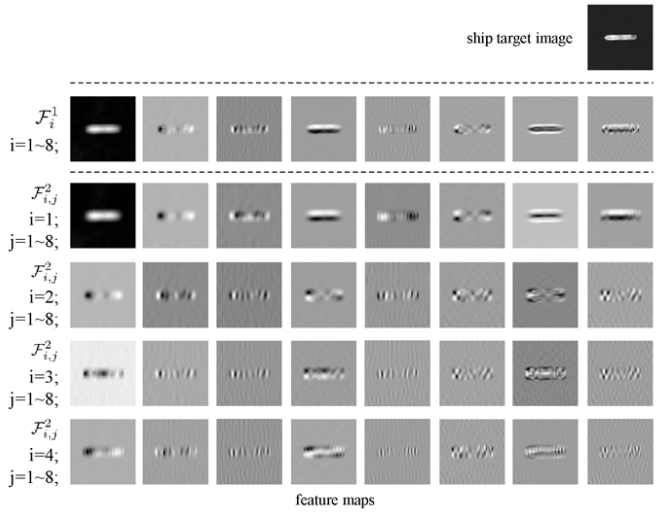


Fig. 6. Ship target sample (1^{st} row) and its feature maps (2^{st} row: C1 layer, $3^{rd}-6^{th}$ rows: C2 layer). In this figure, we set $L_1 = 8$ and $L_2 = 4$.

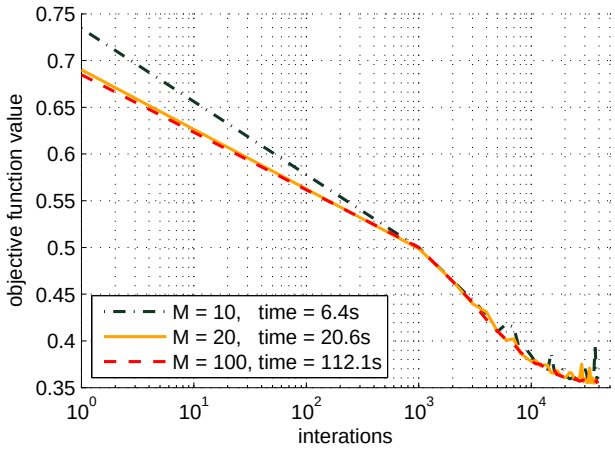


Fig. 7. Training process of C3 layer. The objective function values of different iterations are plotted with different choices of mini-batch number $M = 10$, 20 and 100. Their total training time costs are $6.4s$, $12.6s$ and $112.1s$. The horizontal axis is set as logarithmic scales for a better comparison. In this paper, we set $M = 20$.

filters of C3 that automatically learned by Algorithm 2. We can see the C1 and C2 filters typically cover the most basic image structures of the ship targets, e.g., the edges, stripes and other basic 2D patterns, and the C3 filter typically cover the structural characteristics of the ship target. Fig. 6 shows the corresponding feature maps of a ship target, where the 1^{st} row shows a sample ship target, the 2^{nd} row shows the feature maps of C1 layer $\mathcal{F}_i^1$, $i = 1, \ldots, 8$, and the $3^{rd}-6^{th}$ rows show feature maps of C2 layer $\mathcal{F}_{i,j}^2$, $i = 1, \ldots, 8$, $j = 1, \ldots, 4$. Fig. 8 shows two typical scenes of input images and their corresponding ship probability maps. We can see that SVDNet has successfully suppressed the undesired backgrounds and highlighted all the ship targets including both in-shore and

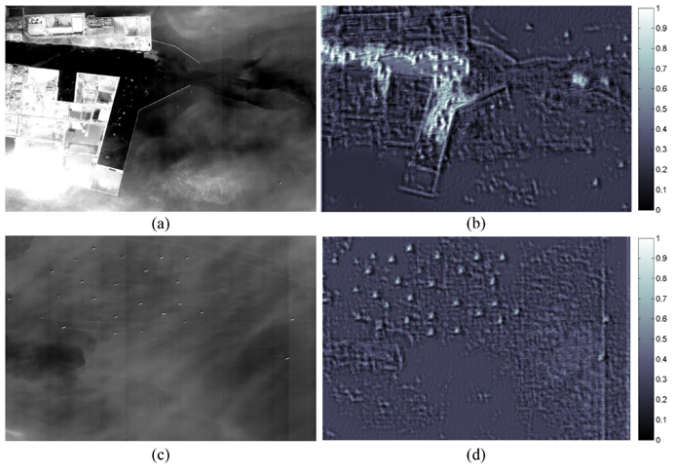


Fig. 8. (a) and (c): Two original image slices. (b) and (d): Corresponding ship probability maps obtained by SVDNet, where a high pixel value suggests a high probability that the filter window covers a ship target.

off-shore ships.

Our detection algorithm is tested on the seven test images of table II. Some of these images have simple and calm sea surface, while other images have waves, clouds and some small islands. The precision and recall rate of the detection results are counted, which are computed as follows

$$\begin{aligned} precision &= N_{tp}/(N_{tp}+N_{fp}), \\ recall &= N_{tp}/(N_{tp}+N_{fn}), \end{aligned} \tag{20}$$

where N_{tp} represents number of true-positives, N_{fp} represents number of false-positives and N_{fn} represents number of false-negatives. Detailed information of the test images and the precision and recall rate are listed in table IV, which suggests an overall high accuracy and stability of our method. In the next two subsections, we will show some ship detection results of complex environments.

TABLE IV
OVERALL SHIP DETECTION RESULT STATISTICS ON SEVEN TEST IMAGES

ID	Waves	Clouds	Islands	Ships	Precision	Recall
2	*No*	*No*	*Yes*	92	0.800	0.869
4	*No*	*No*	*Yes*	16	0.571	0.938
6	*Yes*	*No*	*No*	18	0.833	0.900
8	*Yes*	*Yes*	*No*	15	0.467	0.933
10	*No*	*No*	*No*	23	0.920	0.870
12	*Yes*	*Yes*	*Yes*	26	0.885	0.769
14	*Yes*	*Yes*	*No*	23	0.606	0.869
Avg.	-	-	-	30.4	0.726	0.878

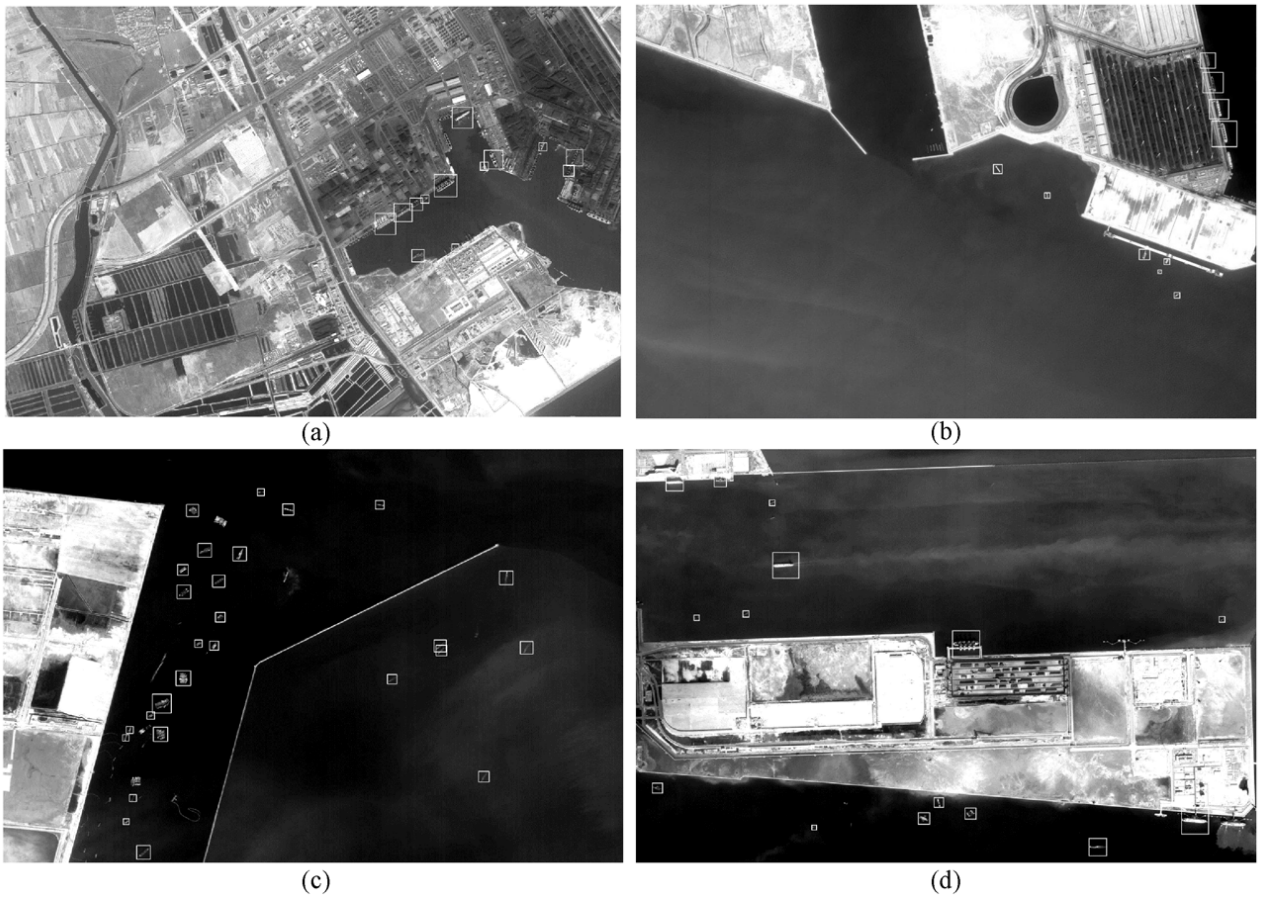


Fig. 9. Some detection results of in-shore ships. Each image slice has the size of 2400×2400 pixels. All slices (original 16-bit images) have been converted to 8-bit images for illustration. It can be seen one ship target is missed in the bottom right image which is adjacent with the land.

C. Detecting In-shore and Off-shore Ships in Complex Environments

In-shore ships and off-shore ships are both very important to ship detection task. Most previous ship detection methods only focus on detecting off-shore ships and ignore in-shore ships. In this subsection, we experiment on some image slices (1600×2400 pixels of each slice) which are cropped near the coastline or around the ports to demonstrate the efficiency of SVDNet. Four typical results are displayed in Fig. 9. We can see most in-shore ships and off-shore ships are successfully detected.

The cloud occlusions and the inference of large waves are other two main challenges of ship detection. In this experiment, we also test on some slices (1600×2400 pixels) of complex environments to demonstrate the robustness of our method. Some of these slices contain clouds and large waves, and some of the others contain both of them. Four typical results are displayed in Fig. 10. Although one ship is missed and there is one false alarm Fig. 10 (d), the performance of our method is still acceptable in these complex environments. Although the ship in the middle of Fig. 10 (d) has low intensity and weak contrary to the background environments, our method still successfully

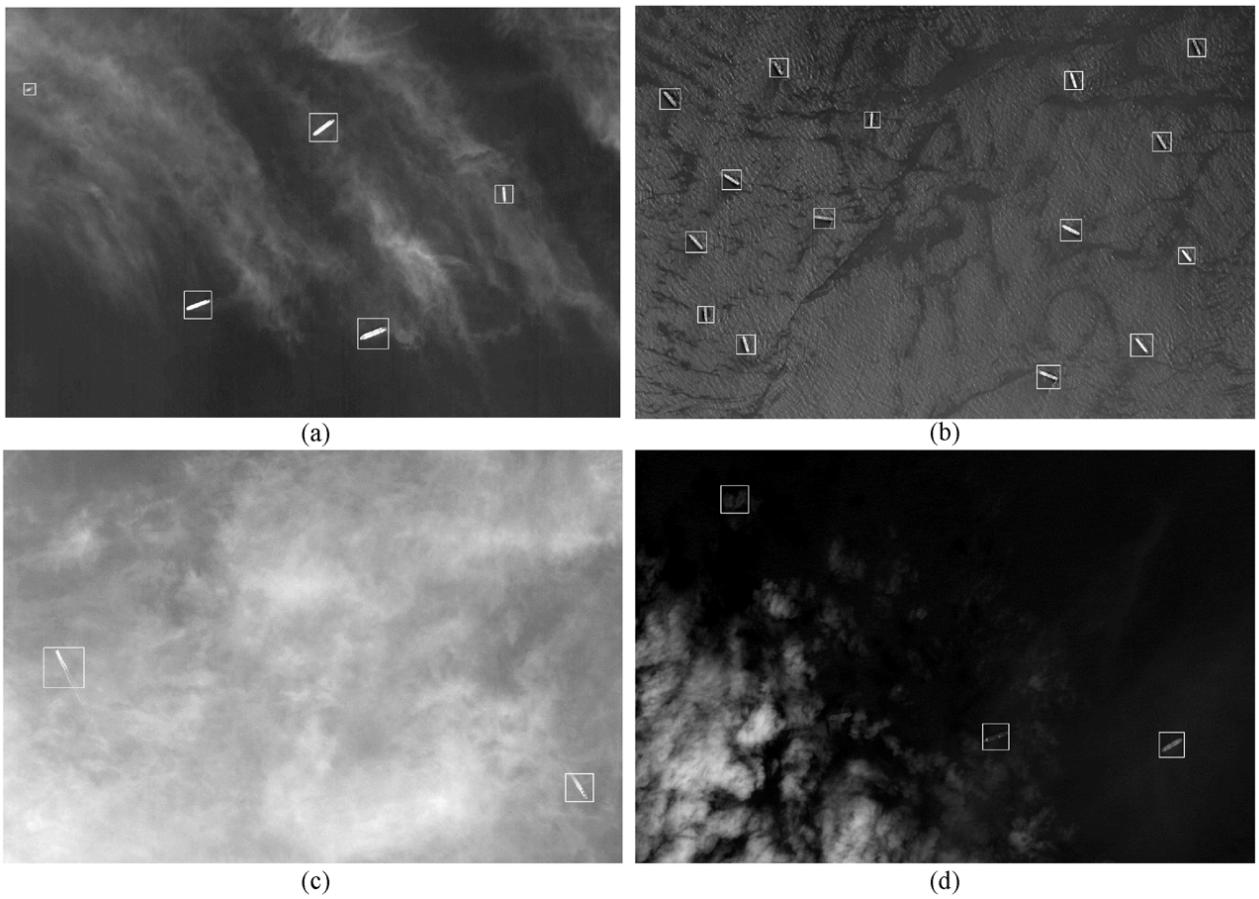


Fig. 10. Some detection results of complex environments such as clouds (a, c, d) and waves (b). Each image slice has the size of 2400×2400 pixels. All slices (original 16-bit images) have been converted to 8-bit images for illustration.

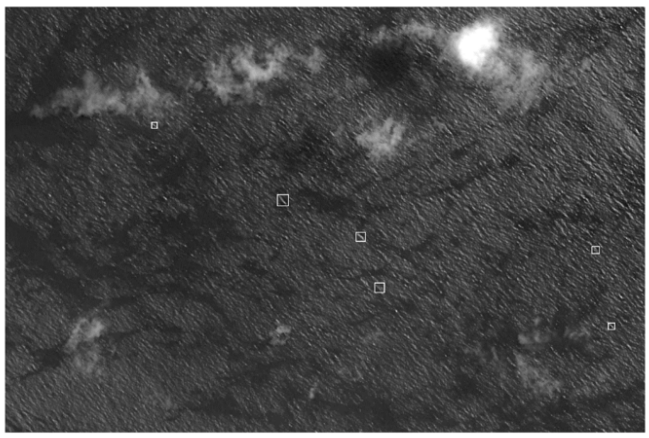


Fig. 11. A failure example of SVDNet in the extreme environments with large waves. There are no ship targets in this slice but our method gives a number of false alarms.

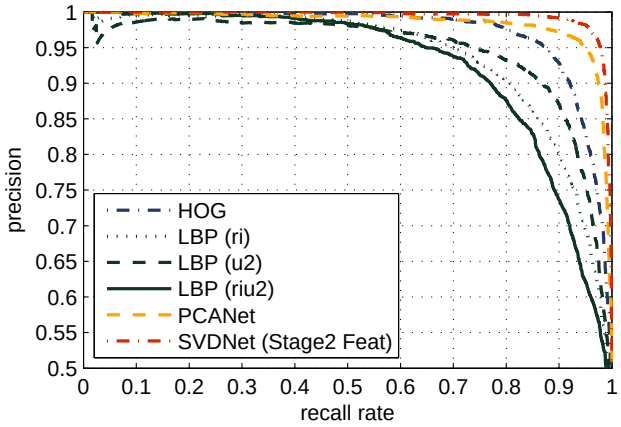


Fig. 12. Precision-recall curves of six different feature descriptors: HOG, LBPri, LBPu2, LBPriu2, PCANet and SVDNet's stage-2 feature.

detect it.

Fig. 11 shows a failure example of SVDNet in extreme environments with large waves. Actually, there are no ships in this slice, but our algorithm gives a number of false alarms. These false alarms resemble the small ships very much in both shape and size, and some of them even can not be easily identified by human eyes. In this condition, if we just take a local view of each candidate, it is still far from discriminating the ship target from the waves. One possible solution of this extreme case is to investigate the surrounding environments of the candidate target, and make use of both local and global information to make the final decision. This will be one of our future works.

V. Experimental Analysis

In this section, we will give a detailed analysis on the efficiency of our SVDNet including feature performance, model complexity, parameter analysis and training set built-up strategy.

A. Compare with Other Features

To further illustrate the effectiveness of our method, we compare our SVDNet's stage-2 feature with some classical feature descriptors including HOG [25] , LBP [23] and PCANet [26]. We use the same parameter settings of HOG, LBP and PCANet with those corresponding ones of our SVDNet, including the same sample size $S = 80$, the same block size $B = 16$. Among these features, LBP has three common types: "LBPri", "LBPu2" and "LBPriu2", which represent "Rotation Invariant LBP", "Uniform LBP" and "Rotation Invariant and Uniform LBP", respectively. We use all the augmented ship samples in this experiment as our dataset. We randomly choose 2/3 samples (8020 positives and 11333 negatives) for training and the rest for test (4010 positives and 5667 negatives). Fig. 12 shows the precision-recall curves of the six types of features: HOG, LBPri, LBPu2, LBPriu2, PCANet and SVDNet's stage-2 feature. We can see SVDNet outperforms than other five kinds of features including PCANet. Although

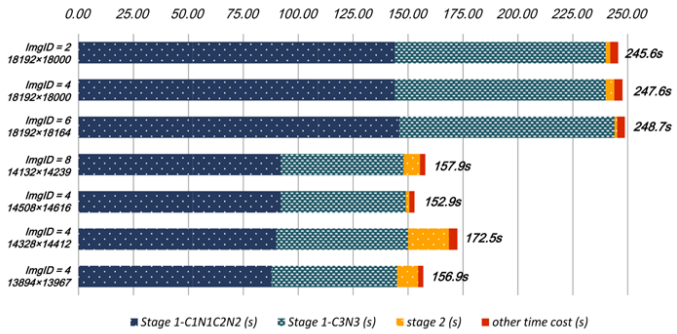


Fig. 13. Computation time of each stage of SVDNet on the 7 test images. Other time cost refers to disk read time and result labeling time and etc.

SVDNet shares some common network structures with PCANet, there are two important differences which make SVDNet a better performance. The first one is the different learning algorithms between SVD and PCA, and the second one is the SVDNets nonlinear mapping operation which adds nonlinearity and stronger representation ability. Despite we do not directly compare our SVDNet with some previous ship detection methods, we believe a detailed comparison with these feature also gives some reflections on their detection performances. Particularly some of these detection methods are build based these features [9, 12, 16]

B. Computational Complexity and Speedup Method

The basic structural elements of SVDNet is simple and computationally efficient. For example, if we have a $H \times W$ panchromatic image and a $k \times k$ convolutional filter, each pixel of the image needs $\mathcal{O}(k^2)$ computations, thus the 2D convolution will have the complexity of $\mathcal{O}(k^2HW)$. The nonlinear mapping layer will have the complexity of $\mathcal{O}(HW)$. Thus in the ship candidate extraction stage, C1 $\sim$ C3 and M1 $\sim$ M3 will totally have the complexity of $\mathcal{O}_1 = \mathcal{O}((k_1^2L_1+(k_2^2+k_3^2)L_1L_2)HW)+\mathcal{O}((L_1+L_1L_2+1)HW)$. In the ship verification stage, each ship candidate window will have the complexity of $\mathcal{O}_2 = \mathcal{O}(L_1L_2S^2 + L_1S^2)$ for feature pooling, and $\mathcal{O}_3 = \mathcal{O}(L_1 2^{L_2-1}S^2/B^2)$ for linear SVM classification. In practice, k,L_1,L_2,S and B are much smaller than the image size H and W, and S and B are in the same order of magnitude. Assume that there are n ship candidates need to be verified in the ship verification stage. In this case, the complexity of the whole detection process becomes

$$\begin{aligned} \mathcal{O}_{SVDNet} &= \mathcal{O}_1 + \mathcal{O}_2 + n\mathcal{O}_3 \\ &= \mathcal{O}(k^2L_1L_2HW + 2^{L_2-1}\frac{L_1S^2}{B^2}n). \end{aligned} \tag{21}$$

In our experiment, the average computational time of each stage of our SVDNet is counted. We test our algorithm on an Intel i7 PC with 16G RAM. The programming language of the detection framework is C++. The programming language of feature learning algorithms is Matlab. Some parallel programming techniques are used to speed up the whole detection process. On ship candidate extraction stage, there are several methods to speed up the convolution

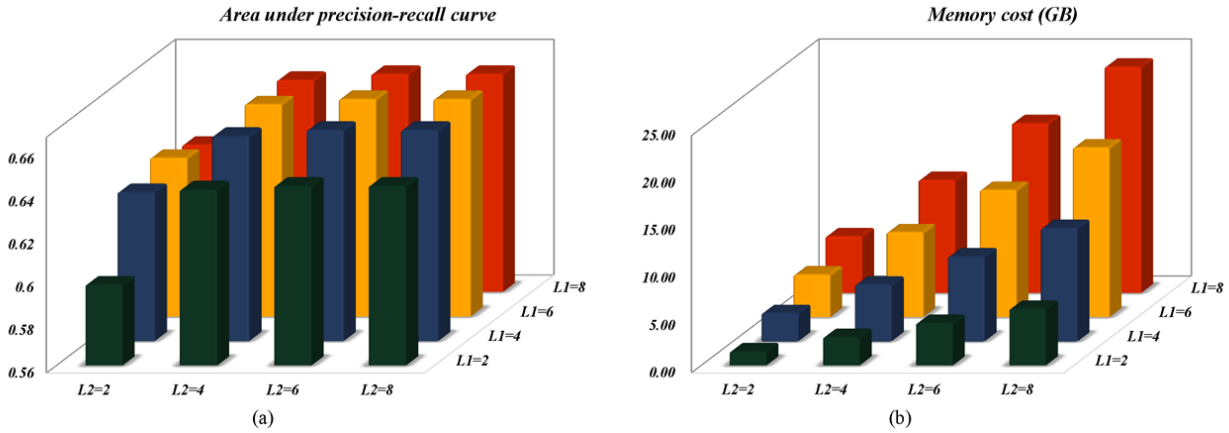


Fig. 14. (a) Area Under the precision recall Curves (AUC) of the final classifier in the ship verification stage with different parameter $L_1, L_2 = \{2, 4, 6, 8\}$, where a higher value of AUC means a better performance. (b) Memory cost of SVDNet for a 10000×10000 sized input image, where we can notice that when $L_1 = L_2 = 8$, the memory cost will reach 23Gb. This is a weakness of our method, which need to be improved in our future works.

operations. For example, the convolution will become much faster if it is applied in Fourier space, since convolution will become the element wise production as $\mathcal{I} \otimes \mathcal{W} \doteq F^{-1}(F(\mathcal{I}) \cdot F(\mathcal{W}))$, where F is the 2D Fast Fourier Transform (2D-FFT). We use the OpenCV Filter Engine to accelerate the 2D convolution operations, and we use the Open-MP toolkits to accelerate the nonlinear mapping operations. On ship verification stage, the feature pooling and the binary classification process for different ship candidates can be paralleled run in several individual threads. In this stage, we also use the open-MP toolkits for parallel operation, which makes the program runs 2-3 times fast than in single thread. Fig. 13 shows the computation time of each stage on the 7 test images. Although Stage-1 takes the most of the detection time, our algorithm still finishes the whole detection pipeline of a 14000×14000 image within 3 minutes. We have also tested on smaller images. Our algorithm takes less than 20 seconds for a 5000×5000 image, and takes less than 10 seconds for a 3000×3000 image, which fully demonstrates the time efficiency of our algorithm.

C. Parameter Analysis

There are three important parameters in SVDNet, the filter number L_1, L_2 and the threshold value η. In this subsection, we made two experiments on L_1, L_2 and η.

In the first experiment, to choose an appropriate value of η , the recall rates are counted with different choices of η, which is shown in Fig. V. We can see $\eta = 0.6$ is a turning point of the recall rate. When η is set larger than 0.6, the recall rate decreases quickly. On the other hand, when η is set smaller than 0.6, it will slow down the following processes and bring additional false positives. In this paper, we choose $\eta = 0.6$. In the second experiment, we test the discriminative ability of the feature pooling process with different choices of L_1 and L_2. Fig. 14 (a) gives an 2D illustration of the linear SVM classification performance on different choices of $L_1, L_2 = \{2, 4, 6, 8\}$.

We randomly choose 1/3 samples for training, and we use the left for test. The scores in Fig. 14 (a) refers to the "Area Under the precision and recall Curves (AUC)". Clearly, a higher AUC means a better performance. When we increase the values of L_1 and L_2, we can observe obvious improvements of the classifier's performance. However, larger L_1 and L_2 also means larger space complexity and computational cost. Fig. 14(b) shows the memory cost (bytes) of the networks for a $10,000 \times 10,000$ sized input image. Since we have to save $L_1 L_2$ full-sized feature maps, the memory cost will become extremely large if we set large values of L_1 and L_2, unless we divide the image in several parts and run our method individually. For example, suppose we use a single precision floating variable to storage each point of the feature map, when $L_1 = L_2 = 8$, the memory cost can reach up to 23Gb. This can be considered as one disadvantage of our method which needs to be further improved in our future works. In this paper, an appropriate choice of L_1 and L_2 would be $L_1 = \{4, 6, 8\}$ and $L_2 = 4$.

TABLE V
SHIP DETECTION RECALL RATES OF THE FIRST STAGE WITH DIFFERENT CHOICES OF η.

η	0	0.2	0.4	0.6	0.8	1.0
recall rate	1.000	0.999	0.994	0.960	0.668	0

D. Knowledge Transfer of Google Samples

In this subsection we have made another experiment on the effect of adding Google training samples. In our method, the uses of the Google samples are summarized as follows:

1) *Knowledge transfer*: Since we have limited GF-1 and VRSS-1 training samples, the Google samples can provide additional information of the ship targets.
2) *Enhance generalization ability*: To improve the adaptability and robustness of different types of remote sensing images.
3) *Data balance*: To balance the costs of positive and negative training samples.

In this experiment, we randomly choose 2/3 of the GF-1 and VRSS-1 samples (2020 augmented samples) for training, and other 1/3 for test (1010 augmented samples). On basis of this, we also test by adding additional Google ship samples (9000 augmented samples) to our training set. Fig. 15 shows some ship samples from GF-1, VRSS-1 and Google images. Fig. 16 shows the precision-recall curves of the above two experiments (with Google samples vs without Google samples). It can be clearly seen that the classification performance has increased by adding Google samples to the training set, which verifies the rationality of our approach.

VI. CONCLUSION AND FUTURE WORKS

We proposed a novel ship detection method for optical satellite image named SVDNet. SVDNet is designed based on the recent popular CNN and the SVD algorithm, which is efficient, structurally compact and theoretically profound. The experiments are made on a set of optical images of GF-1 and VRSS-1 satellite, and the experimental

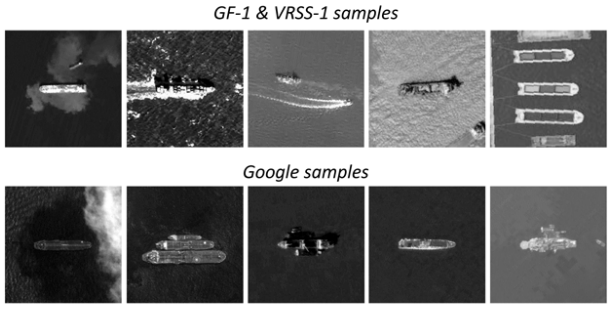


Fig. 15. Some ship samples collected from GF-1 and VRSS-1 (upper part), and Google (lower part) images.

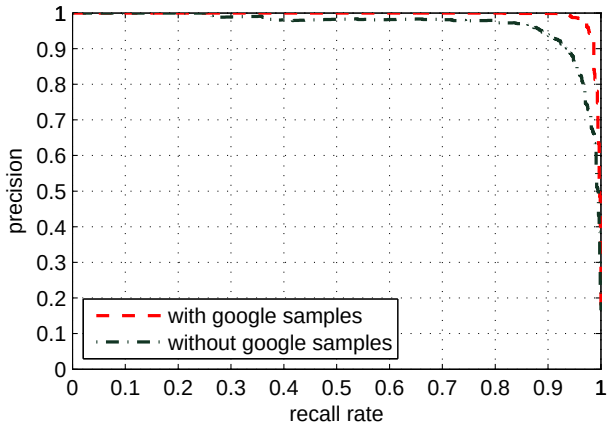


Fig. 16. Precision-recall curves with Google samples and without Google samples.

results demonstrate the superiority of SVDNet. Although there are some improved space of our algorithm, we still consider SVDNet provides a novel and promising ship detection framework, which may give some new cues and directions in ship detection problems and even other related fields. Our future works will focus on the following two aspects. The first aspect is to integrate the information of both local and global environments to improve the detection performance in some extreme environments, and the second aspect is to reduce its memory cost.

REFERENCES

[1] D. J. Crisp, "The state-of-the-art in ship detection in synthetic aperture radar imagery," Aust. Gov., Dept. Defence, Edinburgh, Australia, DSTO-RR-0272, 2004.

[2] F. Y. Lure and Y.-C. Rau, "Detection of ship tracks in AVHRR cloud imagery with neural networks," in *Proc. IEEE IGARSS*, 1994, vol. 3, pp. 1401-1403.

[3] J. M. Weiss, R. Luo, and R. M. Welch, "Automatic detection of ship tracks in satellite imagery," in *Proc. IEEE IGARSS*, 1997, vol. 1, pp. 160-162.

[4] Y. Xia, S. H. Wan, and L. H. Yue, "A novel algorithm for ship detection based on dynamic fusion model of multi-feature and support vector machine," in *Proc. ICIG*, 2011, pp. 521-526.

[5] F. Chen, W. Yu, X. Liu, K. Wang, L. Gong, and W. Lv, "Graph-based ship extraction scheme for optical satellite image," in *Proc. IEEE IGARSS*, 2011, pp. 491-494.

[6] G. Jubelin, and A. Khenchaf, "Multiscale algorithm for ship detection in mid, high and very high resolution optical imagery," in *Proc. IEEE IGARSS*, 2014, pp. 2289-2292.

[7] J. Antelo, G. Ambrosio, J. González alez, and C. Galindo, "Ship detection and recognition in high-resolution satellite images," in *Proc. IEEE IGARSS*, 2009, pp. 514-517.

[8] F. K. Bi, B. C. Zhu, L. N. Gao, and M. M. Bian, "A visual search inspired computational model for ship detection in optical satellite images," *IEEE Geosci. Remote Sens. Lett.*, vol. 9, no. 4, pp. 749-753, Jul. 2012.

[9] S. Qi, J. Ma, J. Lin, Y. Li, and J. Tian, "Unsupervised ship detection based on saliency and S-HOG descriptor from optical satellite images," *IEEE Geosci. Remote Sens. Lett.*, vol. 12, no. 7, pp. 1451-1455, Jul. 2015.

[10] C. Corbane, L. Najman, E. Pecoul, L. Demagistri, and M. Petit, "A complete processing chain for ship detection using optical satellite imagery," *Int. J. Remote Sens.*, vol. 31, no. 22, pp. 5837-5854, Jul. 2010.

[11] C. Corbane, F.Marre, and M. Petit, "Using SPOT-5 HRG data in panchromatic mode for operational detection of small ships in tropical area," *Sensors*, vol. 8, no. 5, pp. 2959-2973, May 2008.

[12] C. Zhu, H. Zhou, R. Wang, and J. Guo, "A novel hierarchical method of ship detection from spaceborne optical image based on shape and texture features," *IEEE Trans. Geosci. Remote Sens.*, vol. 48, no. 9, pp. 3446-3456, Sep. 2010.

[13] F. Bi, F. Liu, and L. Gao, "A hierarchical salient-region based algorithm for ship detection in remote sensing images," in *Advances in Neural Network Research and Applications*. New York, NY, USA: Springer-Verlag, 2010, pp. 729-738.

[14] Z. Ding, Y. Yu, B. Wang, and L. Zhang, "An approach for visual attention based on biquaternion and its application for ship detection in multispectral imagery," *Neurocomputing*, vol. 76, no. 1, pp. 9-17, Jan. 2012.

[15] J. Tang, C. Deng, G.-B. Huang, and B. Zhao, "Compressed-domain ship detection on spaceborne optical image using deep neural network and extreme learning machine," *IEEE Trans. Geosci. Remote Sens.*, vol. 53, no. 3, pp. 1174C1158, Mar. 2015.

[16] Z. Shi, X. Yu, Z. Jiang, and B. Li, "Ship detection in high-resolution optical imagery based on anomaly detector and local shape feature," *IEEE Trans. Geosci. Remote Sens.*, vol. 52, no. 8, pp. 4511-4523, Aug. 2014.

[17] J. Xu, X. Sun, D. Zhang, and K. Fu, "Automatic detection of inshore ships in high-resolution remote sensing images using robust invariant generalized hough transform," *IEEE Geosci. Remote Sens. Lett.*, vol. 11, no. 12, pp. 2070-2074, Dec. 2014.

[18] N. Yokoya, and A. Iwasaki, "Object localization base on sparse representation for remote sensing imagery," in *Proc. IEEE IGARSS*, 2014, pp. 2293-2296.

[19] G. Yang, B. Li, S. Ji, F. Gao, and Q. Xu, "Ship detection from optical satellite images based on sea surface analysis," *IEEE Geosci. Remote Sens. Lett.*, vol. 11, no. 3, pp. 641-645, Mar. 2014.

[20] G. Liu, Y. Zhang, X. Zheng, X. Sun, K. Fu, and H. Wang, "A new method on inshore ship detection in high-resolution satellite images using shape and context information," *IEEE Geosci. Remote Sens. Lett.*, vol. 11, no. 3, pp. 617-621, Mar. 2014.

[21] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: a review and new perspectives," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1798-1828, Aug. 2013.

[22] T. P. Weldon, W. E. Higgins, and D. F. Dunn, "Efficient gabor filter design for texture segmentation," *Pattern Recognit.*, vol. 29, no. 12, pp. 2005-2015, Dec. 1996.

[23] T. Ojala, M. Pietikäinen, and T. Mäenpää, "Multiresolution gray-scale and rotation invariant textur classification with local binary patterns," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 7, pp. 971-987, Jul. 2002.

[24] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vision*, vol. 60, no. 2, pp. 91-110,

Nov. 2004.

[25] N. Dalal, and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. IEEE CVPR*, Jun. 2005, pp. 886-893.

[26] T.-H. Chan, K. Jia, J. Lu, Z. Zeng, and Y. Ma, "PCANet: a simple deep learning baseline for image classification?" *IEEE Trans. Image Process.*, vol. 24, no. 12, pp. 5017-5032, Dec. 2015.

[27] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278-2323, Nov. 1998.

[28] J. Bruna, and S. Mallat, "Invariant scattering convolution networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1872-1886, Aug. 2013.

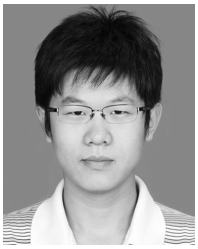
[29] I. Goodfellow, A. Courville, and Y. Bengio, "Deep learning," *Book in preparation for MIT Press.*, [Online]. Available: http://goodfeli.github.io/dlbook/, 2015.

[30] GSHHG - A Global Self-consistent, Hierarchical, High-resolution Geography Database. [Online]. Available: http://www.ngdc.noaa.gov/mgg/shorelines/gshhs.html.

[31] M. M. Cheng, Z. Zhang, W. Y. Lin, and P. Torr, "BING: Binarized normed gradients for objectness estimation at 300fps," in *Proc. IEEE CVPR*, 2014, pp. 3286-3293.

[32] V. N. Vapnik, and V. Vapnik, "Statistical learning theory (Vol. 1)," 1998, New York: Wiley.

[33] R. E. Fan, K. W. Chang, C. J. Hsieh, X. R. Wang, and C. J. Lin, "Liblinear: A library for large linear classification," *J. Mach. Learn. Res.*, vol. 9, pp. 1871-1874, 2008.



Zhengxia Zou received the B.S. degree from the School of Astronautics, Beihang University, Beijing, China, in 2013. He is currently working toward the Ph.D. degree in the Image Processing Center, School of Astronautics, Beihang University. His research interests include pattern recognition and remote sensing target detection and recognition.

Zhenwei Shi (M13) received his Ph.D. degree in mathematics from Dalian University of Technology, Dalian, China, in 2005. He was a Postdoctoral Researcher in the Department of Automation, Tsinghua University, Beijing, China, from 2005 to 2007. He was Visiting Scholar in the Department of Electrical Engineering and Computer Science, Northwestern University, U.S.A., from 2013 to 2014. He is currently a professor and the dean of the Image Processing Center, School of Astronautics, Beihang University. His current research interests include remote sensing image processing and analysis, computer vision, pattern recognition, and machine learning.

He has authored or co-authored over 100 scientific papers in refereed journals and proceedings, including the IEEE Transactions on Pattern Analysis and Machine Intelligence, the IEEE Transactions on Neural Networks, the IEEE Transactions on Geoscience and Remote Sensing, the IEEE Geoscience and Remote Sensing Letters and the IEEE Conference on Computer Vision and Pattern Recognition.