Target Classification Using the Deep Convolutional Networks for SAR Images

Sizhe Chen, *Student Member, IEEE*, Haipeng Wang, *Member, IEEE*, Feng Xu, *Senior Member, IEEE*, and Ya-Qiu Jin, *Fellow, IEEE*

***Abstract*—The algorithm of synthetic aperture radar automatic target recognition (SAR-ATR) is generally composed of the extraction of a set of features that transform the raw input into a representation, followed by a trainable classifier. The feature extractor is often hand designed with domain knowledge and can significantly impact the classification accuracy. By automatically learning hierarchies of features from massive training data, deep convolutional networks (ConvNets) recently have obtained state-of-the-art results in many computer vision and speech recognition tasks. However, when ConvNets was directly applied to SAR-ATR, it yielded severe overfitting due to limited training images. To reduce the number of free parameters, we present a new all-convolutional networks (A-ConvNets), which only consists of sparsely connected layers, without fully connected layers being used. Experimental results on the Moving and Stationary Target Acquisition and Recognition (MSTAR) benchmark data set illustrate that A-ConvNets can achieve an average accuracy of 99% on classification of ten-class targets and is significantly superior to the traditional ConvNets on the classification of target configuration and version variants.**

***Index Terms*—Automatic target recognition (ATR), deep convolutional networks (ConvNets), deep learning, synthetic aperture radar (SAR).**

I. Introduction

AIRBORNE and spaceborne synthetic aperture radar (SAR) can operate in all-weather day-and-night conditions and make high and very high resolution images. SAR is particularly suitable for target classification, reconnaissance, surveillance, etc. Because of scattering/imaging mechanism and speckle in SAR imagery, the interpretation and understanding of SAR images are much different from visual photo analysis. Searching for small targets of interest in massive SAR images by eye is time consuming and often impractical. This suggests the development of automatic target recognition (ATR) algorithms for SAR images.

A standard architecture of SAR-ATR proposed by the MIT Lincoln Laboratory was described as three stages: detection, discrimination, and classification [1]. Detection is to extract candidate targets from SAR images using a constant false alarm rate (CFAR) detector [2]. The output might include not only targets of interests such as tanks, armored vehicles, and missile launchers but also false alarm clutter such as trees, buildings, bridges, and cars. At the following discrimination stage, in order to eliminate false alarms, several features are selected to train a discriminator to solve the two-class (target and clutter) problem [3]. Finally, the classifier is utilized to categorize each input as a specific target type.

There are three mainstream paradigms on the final classification stage: template matching, model-based methods, and machine learning. In the template-based semiautomated image intelligence processing (SAIP) system [4], the mean square error classifier is used to find the best match between the target data and the template database. The classification accuracy of the SAIP system is satisfactory if the target configuration is similar to those in the template database. However, the performance degrades significantly [5] when the targets are in extended operating conditions (EOC) [6], as follows: 1) target configurations variability, e.g., skirts presented along both sides of the target or fuel drums fixed on the rear; 2) its turret is rotated to arbitrary position; 3) background variability, e.g., meadow, road in forest, desert, and urban area; and 4) targets are obscured in revetments.

To solve this problem, a model-based Moving and Stationary Target Acquisition and Recognition (MSTAR) system [7] was developed. The main characteristics of the model-based methods are described as follows: given a SAR image chip, first, a set of hypotheses concerning target category and pose are produced; then, the SAR images of the targets in the hypothesis list are predicted, using target computer-aided design model and electromagnetic simulation software, and finally, a set of predicted features are compared with those extracted from the actual SAR image chips.

With the emergence of several robust trainable classifiers, such as artificial neural network [8], support vector machine (SVM) [9], and AdaBoost [10], the machine learning paradigms have been applied to the SAR-ATR studies. Instead of selecting the best match, the classification problem is solved by first designing a set of different features to represent the targets and then utilizing these feature vectors to train a classifier. By minimizing a cost function, the optimal decision boundaries on a labeled training set can be automatically found. The feature space is partitioned into several independent regions, each of which corresponds to a particular target category. In machine learning approaches, most of the work is to design an appropriate set of feature extractors, since it is often task

Manuscript received May 14, 2015; revised September 6, 2015, January 6, 2016, and March 10, 2016; accepted March 29, 2016. Date of publication April 27, 2016; date of current version June 1, 2016. This work was supported by the National Natural Science Foundation of China under Grant 61571132, Grant 61571134, 61471127, and Grant 61331020.

The authors are with the Key Laboratory of Information Science of Electromagnetic Waves (Ministry of Education), Fudan University, Shanghai 200433, China (e-mail: hpwang@fudan.edu.cn; fengxu@fudan.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TGRS.2016.2551720

specific. The designed features should discriminate among different categories and be robust with respect to translation and intraclass deformation. Similar to the progress in speech and object recognition, much of the performance improvement on SAR-ATR is also attributable to the discovery of better features and feature fusion. Features commonly used in SAR images include geometric descriptors, such as peak locations, edges, corners, and shapes [11], and transform-domain coefficients, such as wavelet coefficients [12].

With recent advances of deep learning theory [13], e.g., in speech and optical image recognition domain, considerable effort has been made to design multistage architectures that automatically learn hierarchical features from data sets. In many benchmark data sets for speech or optical image recognition, these feature learning algorithms have achieved superior performance over hand-designed features. Nowadays, deep convolutional networks (ConvNets) [14] are found to be highly effective in object detection and recognition. A most remarkable result was realized in the 2012 ImageNet data set, where 1.2 million images with 1000 different object categories were included in the training set. On the test data set, the deep ConvNets obtained the error rates of 15.3%, substantially lower than the previous state-of-the-art result (26.2%) [14]. Furthermore, ConvNets have achieved many impressive results on various computer vision tasks, such as handwritten digits [15], traffic sign [16], and face recognition [17].

Successful training ConvNets needs large amounts of data. However, in many cases, the training data are not large enough. A common strategy is to first conduct a supervised pretraining of ConvNets with a large data set and then to fine-tune the network on the task-specific small data set [18]. However, since the number of SAR images for specific targets is often limited, applying the ConvNets directly to SAR-ATR can cause severe overfitting.

Hence, a new architecture of ConvNets with fewer degrees of freedom is proposed. By only using sparsely connected convolution architectures, we present a new all-convolutional networks (A-ConvNets), which is directly applicable to SAR images. All layers of the A-ConvNets take the form of sparse connection (convolution) rather than full connection. A-ConvNets can successfully avoid the overfitting issue associated with small training data sets. The experimental results show that, using data-driven features learned automatically from SAR image pixels, A-ConvNets achieves excellent classification accuracy.

The remainder of this paper is organized as follows. Section II gives an introduction of the overall architecture and components of our A-ConvNets. In Section III, some details of training A-ConvNets are described. Some experimental results on the MSTAR data set are presented in Section IV. Section V gives the conclusions.

II. Guidelines for Manuscript Preparation Architecture of A-ConvNets

Here, we first discuss the characteristic and general layout of our A-ConvNets and then describe the details of components used in A-ConvNets. Finally, we show the specific configuration in our implementation.

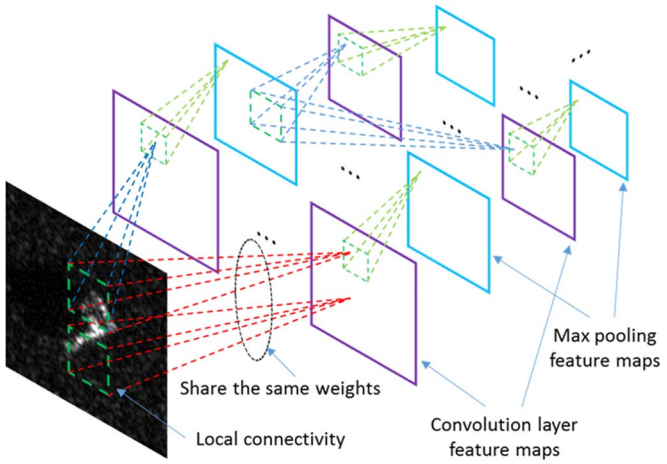

Fig. 1. Illustration of ConvNets, alternating with convolution and pooling layers.

As shown in Fig. 1, the ConvNets are composed of several alternations of convolution and pooling layers, followed by several fully connected layers on the top. However, the amount of training data in our tasks is not enough to train so many parameters without severe overfitting. It is noticed that most of the trainable parameters are included in the fully connected layers. In addition, some experimental results show that the number of hidden layers is critical to the performance of ConvNets [18]. Thus, instead of decreasing the number of trainable parameters via reducing the number of layers, we replace all the fully connected layers with convolutional layers. Although this alteration reduces the representation capacity of the network, it greatly reduces overfitting. The lack of fully connected layers resembles the architecture of Fukushima's neocognitron [34], which is considered as an origin of the ConvNets. However, the training of neocognitron is based on an unsupervised learning algorithm, which is called self-organization, and some reinforcement learning rules. Moreover, the computation model for each unit in neocognitron is much more complex than in ConvNets. Previously, an effective regularization method, which is called "dropout," [19] is used in the fully connected layers to reduce overfitting. Our experimental results show that using dropout regularization in the convolutional layers remains effective.

The architecture of our A-ConvNets consists of five trainable layers (see Fig. 2). It is based on the deep ConvNets, which is a variant of the deep neural network. Deep neural networks contain several layers, all of which are fully connected. Each hidden unit takes as input all the units in the previous layer [20]. In ConvNets, the hidden units of convolution and pooling layers are organized as a set of 2-D arrays, which are called feature maps [21], as shown in Fig. 1. In the convolution layers, each hidden unit receives as input only a local patch of units on the previous layer. The input units in a local patch are multiplied by a weight matrix, and a nonlinear activation function is applied to the result. Each hidden unit can be regarded as a feature detector, because it will output a large value if the specific feature it represents appears in its input. All the units in a feature map are forced to share identical weights, so each feature map detects a specific feature at different positions on

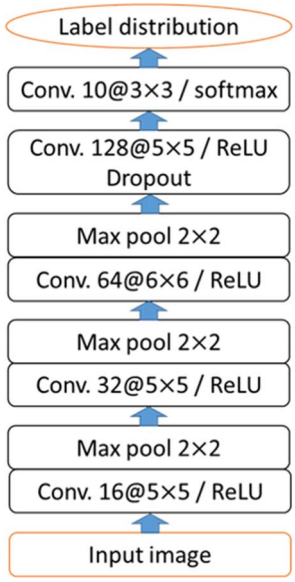

Fig. 2. Overall architecture of the proposed A-ConvNets, composed of five trainable layers. The convolution layers are represented as "conv- (number of feature maps) @ (filter size)."

the previous layer. Due to the use of local connectivity and weight sharing, the number of free parameters to be learned is significantly reduced. The number of feature maps in the subsequent pooling layer is the same as that in the convolution layer. Each pooling unit subsamples a local patch of units on the convolution layer. Typically, a deep ConvNet includes two or more pairs of convolution and pooling layers. The softmax nonlinearity is used in the final output layer when dealing with multiclass classification problems. The details of these operations are described in the following.

A. Convolution

In the convolution layer, the previous layer's input feature maps $O_i^{(l-1)}(i = 1, \ldots, I)$ are connected to all the output feature maps $O_j^{(l)}(j = 1, \ldots, J)$, where $O_i^{(i-1)}(x, y)$ is the unit of the ith input feature map at position (x, y), and $O_j^{(l)}(x, y)$ is the unit of the jth output feature map at position (x, y). Let $k_{ji}^{(l)}(u, v)$ denote the trainable filter (convolution kernel) connecting the ith input feature map to the jth output feature map; $b_j^{(l)}$ denotes the trainable bias of the jth output feature map. Each unit in the convolution layer is computed as

$$O_j^{(l)}(x, y) = f\left(V_j^{(l)}(x, y)\right) \tag{1}$$

$$V_j^{(l)}(x, y) = \sum_{i=1}^{I} \sum_{u,v=0}^{F-1} k_{ji}^{(l)}(u, v) \cdot O_i^{(l-1)}(x - u, y - v) + b_j^{(l)} \tag{2}$$

where $f(x)$ is the nonlinear activation function, and $V_j^{(l)}(x, y)$ denotes the weighted sum of inputs to the jth output feature map at position (x, y). The hyperparameters in the convolution layer include number of feature maps J, filter size $F \times F$, stride S, and zero padding P. The stride specifies the intervals at which to apply the filters to the input feature maps. The convolution operation itself leads to lower dimensional outputs. In order to preserve the spatial size of the feature maps, it is very common to pad the input with zeros on each side of the input. If the input is composed of I feature maps of size $W_1 \times H_1$, then the output is composed of J feature maps of size $W_2 \times H_2$, where $W_2 = (W_1 - F + 2P)/S + 1$, and $H_2 = (H_1 - F + 2P)/S + 1$. The convolution layer, in total, has $I \times J \times F \times F$ weights and J biases, which have to be learned from training data. Recent research has indicated that using small filter sizes (e.g., 3×3 or 5×5) with a stride of 1 usually produces better performance [18]. The number of feature maps on each layer is determined using cross validation. The common setting is that lower layers tend to have fewer feature maps, whereas higher layers tend to have many more. Due to the weight sharing, if the input is shifted, its unit activations will shift the same amount on the output feature map but keep the rest unchanged.

B. Nonlinearity

A nonlinear activation function needs to be added at each convolution layer, since the relationship between the input image and the output label should be a highly nonlinear mapping. In traditional ConvNets [15], a hyperbolic tangent function $f(x) = \tanh(x)$ or a sigmoid function $f(x) = 1/(1 + \exp(-x))$ is applied to each unit on the output feature maps of the convolution layer. The problem is that the gradient of a hyperbolic tangent function tends to zero when they saturate, i.e., when its output is near either -1 or 1, and that stops learning. It used to be hard and cumbersome to train deep models due to these saturating nonlinearities. Recent research has found a nonsaturating nonlinearity, i.e., the rectified linear unit (ReLU) [22], which often works better in practice. The ReLU activation function is given by

$$f(x) = \max(0, x). \tag{3}$$

ReLU achieves a considerable reduction in training time [14]. Moreover, deep networks using ReLU can reach their best performance with purely supervised training on large labeled data sets, without requiring any unsupervised pretraining [22].

C. Pooling

The max pooling operation is applied to each feature map separately [21]. It outputs the maximum value on a group of units located within a local patch, the size of which is called the pooling size. This operation will achieve a small amount of shift and distortion invariance [20]. If the input shifts a small amount, most of the pooling unit values will not change. The max pooling operation is defined as

$$O_i^{l+1}(x, y) = \max_{u,v=0,\ldots,G-1} O_i^{(l)}(x \cdot s + u, y \cdot s + v) \tag{4}$$

where G is the pooling size, and s is the stride, which determines the intervals between neighbor pooling windows. There

are only two kinds of hyperparameter setting found in practice: pooling size 2×2 with a stride of 2 or pooling size 3×3 with a stride of 2. Larger pooling size generally results in worse performance, since it throws away too much information during subsampling.

D. Softmax

The softmax nonlinearity is applied to the output layer on the case of multiclass classification. It will output the posterior probabilities over each class. In other words, the final output is a K-dimensional vector, each element of which corresponds to the probability $p_i = P(y = i|x)$, for $i = 1, \ldots, K$. The softmax nonlinearity operation takes the form [23]

$$p_i = \frac{\exp\left(O_i^{(L)}\right)}{\sum_{j=1}^{K} \exp\left(O_j^{(L)}\right)} \tag{5}$$

where $O_j^{(L)}$ denotes the weighted sum of inputs to the jth unit on the output layer, which is computed using (2). Given a labeled training set of m examples, $\{(x^{(i)}, y^{(i)}), i = 1, \ldots, m\}$, where $y^{(i)}$ refers to the true label; the cross-entropy loss function is defined as

$$L(w) = -\frac{1}{m} \sum_{i=1}^{m} \log P\left(y^{(i)}|x^{(i)}; w\right). \tag{6}$$

By minimizing this loss function, the trainable parameters w will be adapted to increase the probability of the correct class label, which is equivalent to maximum-likelihood estimation. Instead of minimizing squared error, it minimizes the cross entropy between the correct label distribution and probability distribution estimated by the network, which often works better for classification problems. In practice, a regularization term $\lambda\|w\|^2$ is always added to the loss function to prevent overfitting. This regularizer is called L^2 weight decay because adding the L^2 norm to the loss function avoids large values of weights.

E. Dropout

Dropout [19] is an extremely effective and recently proposed regularization technique to reduce overfitting. A good way to improve performance is to combine the predictions produced by a large number of different network architectures. However, it is computational expensive to separately train many networks and average their outputs. Dropout provides a very efficient way to approximately average exponentially many different network architectures. While training, it is implemented by randomly setting the output of each hidden unit to zero with a probability of 0.5. For each training case, it randomly samples a different architecture and only updates the parameters of the sampled network. However, all these sampled networks are very strongly regularized because they share the parameters. At test time, all the hidden units are retained, but their outgoing weights are multiplied by 0.5. This is a good approximation to take the geometric mean of predictions produced by all these sampled networks. The number of trainable parameters in the convolution layers is much smaller than in the fully connected layers; thus, its capacity to overfit has been reduced. In addition, dropout increases training time. For these reasons, dropout is typically used in the fully connected layers. However, adding dropout in the convolution layers as well improves the performance. Moreover, training time in this work is not a problem, because both the network architecture and the amount of training data are relatively small. Thus, we also use dropout in the higher convolution layers to reduce overfitting.

F. Configuration Specifics

The overall architecture of the proposed A-ConvNets in this study is depicted in Fig. 2, which is composed of five convolution layers and three max pooling layers. Each of the first three convolution layers is followed by a max pooling layer, with a pooling size of 2×2 and a stride of 2 pixels. The ReLU nonlinearity is applied to every hidden convolution layer. No spatial zero padding is used in the convolution layer, and the convolution stride is fixed to 1 pixel. The 88×88 input image was filtered by 16 convolution filters of size 5×5 in the first convolution layer, resulting in 16 feature maps of size 84×84. After the first pooling layer, their sizes become 42×42. The first pooling layer's outputs are sent into the second convolution layer, which has a convolution filter size of 5×5, leading to 32 feature maps of size 38×38. After the second pooling layer, their sizes become 19×19. The filter size of the third convolution layer is 6×6, producing 64 feature maps of size 14×14, which becomes 7×7 after pooling. The fourth convolution layer includes 128 feature maps with a convolution filter size of 5×5, which brings out 128 feature maps of size 3×3. The dropout regularization technique is used in this layer. The fifth convolution layer contains ten output units with a convolution filter size of 3×3 to ensure the final output size to be $1 \times 1 \times 10$, each element of which corresponds to the probability for each class.

III. Learning of A-ConvNets

As in most machine learning algorithms, all the weights and biases are learned via minimizing a loss function. Usually, it is not possible to analytically compute the global minimum of the loss function, whereas the gradient of loss function with respect to parameters $\partial L/\partial w$ can be computed analytically. Therefore, the loss function can be minimized through iterative numerical optimization approach. The simplest of such optimization algorithm is the gradient descent. The iterative updating rule for the parameter is $w \leftarrow w - \epsilon(\partial L/\partial w)$, where ϵ is a scalar constant, which is called learning rate. In A-ConvNets, the derivative of loss function with respect to trainable weights on each layer can be efficiently computed using error backpropagation algorithm. However, some special modifications of the general backpropagation algorithm are needed to take into account local connections, weight sharing, and max pooling constraints in the network [24]. Our implementation has referred to the public available code provided by Jia *et al.* [25].

A. Backpropagation in ConvNets

The backpropagation algorithm first computes an intermediate quantity, which is called "error term," i.e., δ_i^l, on each unit and then relates it to the partial derivatives $\partial L/\partial w$. For units in output layer, the error term can be computed as the difference between the actual target value and the network's prediction, i.e.,

$$\delta_i^L = -(y_i - p_i). \tag{7}$$

Then, the error term can be computed backward to lower layers. If the layer $l+1$ is a convolution layer, the error term in the layer l is computed as the cross correlation with zero padding between the convolution kernel $k_{ji}^{(l+1)}$ and error map $\delta_j^{(l+1)}$ that connects to the feature map i in layer l, i.e.,

$$\delta_i^{(l)}(x,y) = \sum_j \sum_{u,v=0}^{F-1} k_{ji}^{(l+1)}(u,v) \cdot \delta_j^{(l+1)}(x+u, y+v). \tag{8}$$

Although there are no trainable weights in the pooling layer, the error terms need to be back propagated to lower layers. If the layer $l+1$ is a max pooling layer, only the unit having the largest values within a group of pooled units receives error term, and the error terms on other units are defined as zero, i.e.,

$$\delta_i^{(l)}(x,y) = f'\left(V_i^{(l)}(x,y)\right) \cdot \sum_{m,n} \delta_i^{(l+1)}(m,n) \cdot \varepsilon(u_{i,m} + m \cdot s - x, v_{i,n} + n \cdot s - y) \tag{9}$$

where $f'(x)$ is the derivative of nonlinear activation function. In the case of ReLU, its derivative is 1 when $x > 0$; otherwise, it is zero. In addition, $\varepsilon(x,y)$ is the Dirac function, whose value is 1 only if both x and y are equal to 0; otherwise, it is zero. $(u_{i,m}, v_{i,n})$ is the index of the unit having the largest value within a group of pooled units, i.e.,

$$(u_{i,m}, v_{i,n}) = \underset{u,v=0,\ldots,G-1}{\arg\max} O_i^{(l)}(m \cdot s + u, n \cdot s + v). \tag{10}$$

After computing the error term on each layer, the partial derivative of loss function with respect to convolution kernels and bias is computed as follows:

$$\frac{\partial L}{\partial k_{ji}^{(l)}(u,v)} = \sum_{x,y} \delta_j^{(l)}(x,y) \cdot O_j^{(l-1)}(x-u, y-v) \tag{11}$$

$$\frac{\partial L}{\partial b_j^{(l)}} = \sum_{x,y} \delta_j^{(l)}(x,y). \tag{12}$$

B. Minibatch Stochastic Gradient Descent With Momentum

For full-batch gradient decent, all the training examples are used to compute the gradient of loss in each iteration. Because even the true gradient direction is only the steepest descent locally but not the right direction toward the global minimum, it is not necessary to estimate the gradient precisely. Instead, doing updates more frequently helps to search more and faster in parameters space. Thus, a small subset of training data is used to assess the gradient in each iteration. This version is called minibatch gradient decent. In each minibatch, the number of training examples on each class should be approximately equal [26].

Momentum [27] is an approach that can increase the speed of learning, because it reduces oscillations in directions of high curvature by combining gradients with opposite signs and accumulates speed in directions with consistent gradient signs. Instead of using the current gradient to update the weights, the momentum method uses it to change the velocity parameter v_i and then updates the weights according to the current velocity.

The whole network is trained purely supervised using stochastic gradient decent with a minibatch size of 100 examples, combined with a momentum parameter of 0.9 and a weight decay parameter of 0.004. The update rule for weights w is

$$v_{i+1} = 0.9 \cdot v_i - 0.004 \cdot \epsilon \cdot w_i = \epsilon \cdot \left\langle \frac{\partial L}{\partial w_i} \right\rangle_i \tag{13}$$

$$w_{i+1} = w_i + v_{i+1} \tag{14}$$

where i is the number of iteration, ϵ is the learning rate, v is the velocity parameter defined in the momentum method, and $\langle \partial L/\partial w_i \rangle_i$ is the average of the gradient of loss function with respect to w_i, which is computed by training examples on the ith minibatch [14].

C. Weight Initialization

For most deep ConvNets, weights are initialized from Gaussian distributions with zero mean and a standard deviation of 0.01, and biases are initialized with a small constant value of 0.1 [18]. It is important to initialize weights randomly, because if all the weights are initialized to be the same, then each unit will compute the same output value, the same gradients during backpropagation, and, hence, the same weight updates. The latest research suggests that deep networks with ReLU nonlinearity should be initialized by zero-mean Gaussian distributions with a standard deviation of $\sqrt{2/n}$, where n is the number of inputs to each unit [36].

D. Learning Rate

It is usually advantageous to reduce the learning rate during training. Initially, it would be best to use a large learning rate to change the parameters faster. However, with a large learning rate, the parameters will eventually oscillate randomly, unable to converge to a better value. Typically, the initial learning rate is on the order of 10^{-2} or 10^{-3}, whose value is set as the largest one that could decrease the loss function. A common heuristic of learning rate schedule is to watch the validation accuracy during training and tune down the learning rate by a factor of 0.1 or 0.5, whenever the validation accuracy stops improving for some amount of time [26]. In this paper, the learning rate is initially 0.001 and is reduced by a factor of 0.1 after 50 epochs, where epoch denotes the number of times each example has been used during training.

Fig. 3. Types of military targets: (top) optical images versus (bottom) SAR images.

E. Early Stopping

When training large models, it is common to observe that the validation accuracy initially improves steadily over time, but getting worse in the end. Early stopping is a form of regularization technique to avoid going beyond the balance point where underfitting transits to overfitting. A common implementation approach is to store the parameter setting during training whenever there is an improvement on the validation accuracy and, finally, return back to the parameter setting that produces the best validation accuracy [26].

IV. Experiments on MSTAR Data Set

The experiment data used in this paper were collected by the Sandia National Laboratory SAR sensor platform. The collection was jointly sponsored by the Defense Advanced Research Projects Agency and the Air Force Research Laboratory as part of the MSTAR program [7]. Hundreds of thousands of SAR images containing ground targets were collected, including different target types, aspect angles, depression angles, serial number, and articulation, and only a small subset of which are publicly available on the website [28]. The publicly released data sets include ten different categories of ground targets (armored personnel carrier: BMP-2, BRDM-2, BTR-60, and BTR-70; tank: T-62, T-72; rocket launcher: 2S1; air defense unit: ZSU-234; truck: ZIL-131; bulldozer: D7). They were collected using an X-band SAR sensor, in a 1-ft resolution spotlight mode, full aspect coverage (in the range of 0° to 360°). The MSTAR benchmark data set is widely used to test and compare the performance of SAR-ATR algorithms. Examples of SAR images of ten types of targets at similar aspect angles and their corresponding optical images are depicted in Fig. 3. In order to comprehensively assess the performance, the algorithm is tested both under standard operating conditions (SOC) and EOC. SOC refers to that the serial numbers and target configurations in the test set are the same as those in the training set, but with different aspects and depression angles. In EOC test scenarios, there are large differences between the training and test sets, including substantial variations in depression angle, target articulation, and version variants.

TABLE I
Number of Training and Testing Images for the SOC Experimental Setup

		Train		Test	
Class	Serial No.	Depression	No. Images	Depression	No. Images
BMP-2	9563	17°	233	15°	196
BTR-70	c71	17°	233	15°	196
T-72	132	17°	232	15°	196
BTR-60	k10yt7532	17°	256	15°	195
2S1	b01	17°	299	15°	274
BRDM-2	E-71	17°	298	15°	274
D7	92v13015	17°	299	15°	274
T-62	A51	17°	299	15°	273
ZIL-131	E12	17°	299	15°	274
ZSU-234	d08	17°	299	15°	274

A. Results Under SOC

In this SOC experimental setup, the proposed algorithm is evaluated on the ten-target classification problem. The serial number, depression angle, and number of images available for training and testing are listed in Table I. The same target class in the training and testing sets has the same serial number, but differs in azimuth and depression angle. Images for training are captured at 17° depression angle, and images for testing are acquired at 15° depression angle. The number of images per class is not equal. Data augmentation is a commonly used technique that could help improve performance [14]. In this paper, it is implemented by randomly sampling a number of 88×88 patches from the original 128×128 SAR image chips. There is only one target located at the center of the original 128×128 image chip, which ensures that every extracted patch contains a complete target. After this operation, the number of training images per class could be increased at maximum by $(128 - 88 + 1) \times (128 - 88 + 1) = 1681$ times. For example, BMP-2, which originally has 233 images, could get at maximum 233×1681 images by shifting. In this paper, a subset of 2700 images with random shifting is used for each class for training purpose. The model is trained on this artificially expanded data set. No image preprocessing algorithm is applied to the SAR images. Table II shows the confusion matrix of the SOC test scenario. Each row in the confusion matrix denotes the actual target class, and each column represents the class predicted by the network.

In order to test the antinoise performance of the A-ConvNets, we randomly select a certain proportion of pixels in the test image and replace their values with samples generated from a uniform distribution, as shown in Fig. 4. This is consistent with the noise simulation approach in [32]. With the network previously trained on the ten-target classification problem, the accuracies across the level of noise are shown in Table III.

It is shown in Table III that the accuracy decreases with increasing noise level; when the noise level reaches 5%, the accuracy degrades by about 10%. Therefore, to improve the accuracy, it is better to perform noise filtering first.

TABLE II
CONFUSION MATRIX FOR THE SOC EXPERIMENTAL SETUP

Class	BMP-2	BTR-70	T-72	BTR-60	2S1	BRDM-2	D7	T62	ZIL131	ZSU234	P_{cc}(%)
BMP-2	194	0	1	0	1	0	0	0	0	0	98.98
BTR-70	0	195	0	0	0	1	0	0	0	0	99.49
T-72	0	0	196	0	0	0	0	0	0	0	100
BTR-60	1	0	0	188	0	0	0	1	1	4	96.41
2S1	0	0	0	0	269	4	0	0	0	1	98.18
BRDM-2	0	0	0	0	0	272	0	0	0	2	99.27
D7	0	0	0	0	0	0	272	1	1	0	99.27
T-62	0	0	0	0	0	0	0	272	1	0	99.64
ZIL-131	0	0	0	0	0	0	0	0	273	1	99.64
ZSU-234	0	0	0	0	0	0	1	0	0	273	99.64
Total											99.13

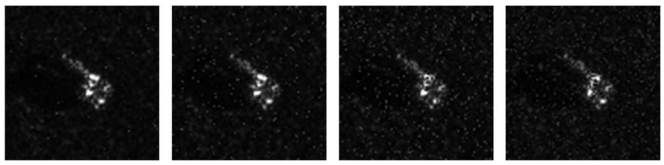

Fig. 4. Illustration of random noise. The levels of noise are 1%, 5%, 10%, and 15%, respectively.

TABLE III
ACCURACIES ACROSS THE LEVEL OF NOISE

Noise	1%	5%	10%	15%
A-ConvNets	0.9176	0.8852	0.7584	0.5468

The internal state of a trained A-ConvNet is depicted in Fig. 5, where a target T72 is forward propagated through a trained network, and the intermediate values of all the feature maps together with a subset of convolution kernels are plotted here. For example, Conv. Layer 2 should have, in total, 32×16 kernels, but we only show the first 64 kernels due to space limitations.

B. Results Under EOC

The proposed algorithm is then evaluated with respect to large depression angle changes. It is well known that SAR images are extremely sensitive to the variance of depression angles. As listed in Table IV, only four types of targets (2S1, BRDM-2, T-72, and ZSU-234) in the MSTAR data set include the examples obtained at 30° depression angle; thus, we test the algorithm on these examples, with the training set chosen from the corresponding four targets in Table I. The confusion matrix for large depression angle variations (denoted by EOC-1) is shown in Table V.

In another EOC test scenario, the algorithm is evaluated with respect to target configuration and version variants. In this experimental setup, the training set is composed of four targets (BMP-2, BRDM-2, BTR-70, and T-72) in 17° depression angles chosen from Table I. There are two groups of test set as listed in Table VI (EOC-2) and Table VII (EOC-2), containing two version variants of BMP-2 and ten version variants of T-72, acquired at both 17° and 15° depression angles. The serial number of targets in the test set is not included in the training set. The confusion matrices for EOC-2 are shown in Tables VIII and IX. Tables VIII and IX suggest that the T-72 and BMP-2 targets are most often confused with one another. However, results of several other methods given in [3] and [31] do not show such apparent trends, in which the misclassified T-72 targets are classified as BMP-2, BRDM-2, and BTR-70 with similar probabilities. However, it is the authors' opinion

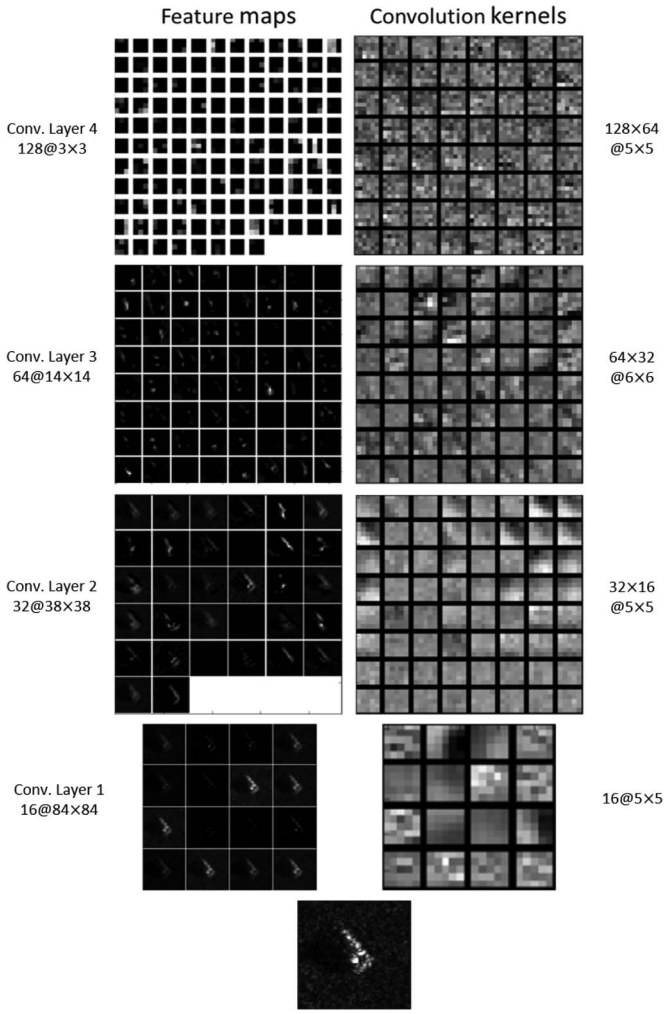

Fig. 5. (Left) Output feature maps and (right) convolution kernels on each layer.

TABLE IV
TEST EXAMPLES FOR EOC-1 (LARGE DEPRESSION VARIATION)

Class	Serial No.	Depression	No. Images
2S1	b01	30°	288
BRDM-2	E-71	30°	287
T-72	A64	30°	288
ZSU-234	d08	30°	288

TABLE V
CONFUSION MATRIX FOR EOC-1 (LARGE DEPRESSION VARIATION)

Class	2S1	BRDM-2	T-72	ZSU-234	P_{cc}(%)
2S1	274	12	1	1	95.14
BRDM-2	0	285	0	2	99.30
T-72	2	1	266	19	92.36
ZSU-234	0	0	4	284	98.61
Total					96.12

TABLE VI
TEST EXAMPLES FOR EOC-2 (CONFIGURATION VARIANTS)

Class	Serial No.	Depression	No. Images
T-72	S7	15°, 17°	419
	A32	15°, 17°	572
	A62	15°, 17°	573
	A63	15°, 17°	573
	A64	15°, 17°	573

TABLE VII
TEST EXAMPLES FOR EOC-2 (VERSION VARIANTS)

Class	Serial No.	Depression	No. Images
BMP-2	9566	15°, 17°	428
	c21	15°, 17°	429
T-72	812	15°, 17°	426
	A04	15°, 17°	573
	A05	15°, 17°	573
	A07	15°, 17°	573
	A10	15°, 17°	567

TABLE VIII
CONFUSION MATRIX FOR EOC-2 (CONFIGURATION VARIANTS)

Class	Serial No.	BMP-2	BRDM-2	BTR-70	T-72	P_{cc}(%)
T-72	S7	7	0	0	412	98.33
	A32	11	0	0	561	98.08
	A62	1	0	0	572	99.83
	A63	6	0	0	567	98.95
	A64	4	0	0	569	99.30
Total						98.93

TABLE IX
CONFUSION MATRIX FOR EOC-2 (VERSION VARIANTS)

Class	Serial No.	BMP-2	BRDM-2	BTR-70	T-72	P_{cc}(%)
BMP-2	9566	412	1	2	13	96.26
	c21	423	1	0	5	98.60
T-72	812	16	0	0	410	96.24
	A04	6	2	0	565	98.60
	A05	1	0	0	572	99.83
	A07	3	0	0	570	99.48
	A10	0	0	0	567	100
Total						98.60

that T-72 looks indeed more like BMP-2 than the rest do, because both T-72 and BMP-2 have circular turret and gun barrel. Thus, in that sense, the results of our method appear to be reasonable.

The performance of A-ConvNets is compared with four widely cited methods and four recently proposed methods. These methods include extended maximum average correlation height (EMACH) [29], SVM [9], conditional Gaussian model (Cond Gauss) [30], AdaBoost [10], iterative graph thickening (IGT) [31], sparse representation of monogenic signal (MSRC) [32], monogenic scale space (MSS) [33], and modified polar mapping classifier (M-PMC) [3]. The average classification accuracies of these algorithms are compared in Table X.

TABLE X
CONFUSION MATRIX FOR AVERAGE CLASSIFICATION ACCURACIES. A-CONVNETS VERSUS STATE-OF-THE-ART METHODS

Method	SOC (%)	EOC-1 (%)	EOC-2 (%)
EMACH [31]	88	77	68
SVM [31]	90	81	75
AdaBoost [31]	92	82	78
Cond Gauss [30]	97	80	79
IGT [31]	95	85	80
MSRC [32]	93.6	98.4	-
MSS [33]	96.6	98.2	-
M-PMC [3]	98.8	-	97.3
A-ConvNets	99.13	96.12	98.93

Note that the results are cited from the corresponding papers. The classification results of EMACH, SVM, and AdaBoost are quoted from [31, Tables III–XXII] (derived from its, because the original papers of EMACH, SVM, and AdaBoost only reported the three-target classification results, without ten-target classification and EOC results being reported. The authors of [31] have implemented these algorithms and published the code online.

C. Results on Outlier Rejection

An evaluation of the confuser rejection performance is conducted by training a network on three target classes (BMP-2, BTR-70, and T-72), with two confuser targets (2S1 and ZIL-131) to be rejected included in the test set. In this experimental scenario, the test set is composed of 588 images of three known targets (196 images per class) and 548 images of two confuser targets (274 images per class). Since the last layer's output is a vector of probabilities, each element of which provides the posterior probability for each class, the confuser rejection rule can be set that if all the posterior probabilities are lower than a threshold value $\tau_{\rm th}$, the target image will be declared as a confuser. After training, the receiver operating characteristic (ROC) curves that describe the detection ratio P_d versus the false alarm ratio $P_{\rm fa}$ can be obtained by altering this threshold value continuously, as shown in Fig. 6, in which P_d is defined as the ratio between the number of known targets detected and the number of known targets in the test set, and $P_{\rm fa}$ is defined as the ratio between the number of confusers declared as known and the number of confusers in the test set. Larger threshold value leads to lower P_d and $P_{\rm fa}$, and smaller threshold value results in higher P_d and $P_{\rm fa}$. As shown in Fig. 6, at the detection ratio $P_d = 90\%$, the false alarm ratio $P_{\rm fa}$ of A-ConvNets is 35.9%. In [35], using the same experimental scenario, at $P_d = 90\%$, the $P_{\rm fa}$ of four SAR-ATR algorithms (template matching, principal component analysis with multiresolution method, neural network with quadratic mutual information cost function, and SVM with Gaussian kernels) are equal to 46.5%, 40%, 45.5%, and 31.2%, respectively. The outlier rejection performance of A-ConvNets is comparable with the results presented in [35].

D. Performance Comparison: A-ConvNets Versus Traditional ConvNets

In order to compare the performance between the A-ConvNets and the traditional ConvNets, four networks with different

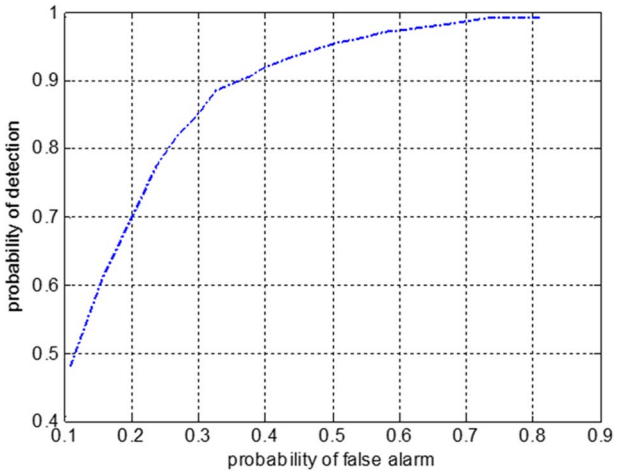

Fig. 6. ROC curves. P_d versus P_{fa}.

TABLE XI
TRADITIONAL CONVNET CONFIGURATIONS, ONE PER COLUMN. THE FULLY CONNECTED LAYERS ARE REPRESENTED AS "FC-(NUMBER OF UNITS)." THE CONVOLUTION LAYERS ARE REPRESENTED AS "CONV-(NUMBER OF FEATURE MAPS) @ (FILTER SIZE)"

Network Configuration			
A	B	C	D
conv-16@5 × 5 + max-pooling			
conv-32@5 × 5 + max-pooling			
conv-64@5 × 5 + max-pooling			
FC-512	FC-1024	FC-512 FC-512	FC-1024 FC-1024
softmax			

TABLE XII
AVERAGE CLASSIFICATION ACCURACIES FOR FIVE DIFFERENT NETWORKS

Network	SOC (%)	EOC (%)
A	97.46	65.12
B	98.03	76.20
C	97.33	71.85
D	97.88	77.84
A-ConvNets	99.13	87.40

architectures are evaluated, as listed in Table XI. The architectures of these networks are similar to that of the A-ConvNets used in this work. The only difference exists in the higher layers: from one fully connected layer with 512 units in the network "A" to two fully connected layers each with 1024 units in the network "D." Dropout regularization method is applied to all the fully connected layers to reduce overfitting. The performance of these four ConvNets is evaluated under both SOC and EOC experimental setups. In the SOC test scenario, the training and testing sets include ten target classes listed in Table I. In the EOC test scenario, the training set is the same as before, but the testing set is composed of all the BMP-2 and T-72 version variants listed in Tables VI and VII. The average classification accuracies for five different architectures are compared in Table XII. Under SOC experimental setup, all these four ConvNets perform well. However, under EOC experimental setup, the performance of the A-ConvNets is significantly superior to the ConvNets with fully connected layers. Hence, we conclude that A-ConvNets is able to significantly suppress overfitting when training data are scarce and, thus, suitable for SAR-ATR.

E. End-to-End SAR-ATR Cases

The preceding studies involve only the target classification stage of SAR-ATR, where SAR image chips are used as input to the classifier. An end-to-end SAR-ATR system would have to first detect potential targets and isolate the regions out from a complex background, for example, forest, urban area, and sea surface. The second stage is to feed these isolated image chips to a classifier and ultimately declare the recognized target type. Target detection could be implemented via conventional CFAR-type algorithms. In this paper, in order to present such an end-to-end case, we choose to use two stages of A-ConvNets, with the first one performing binary classification, i.e., detection, and the second performing recognition.

- *Data Preparation:* The publicly available MSTAR data set includes a large number of scene images of size 1748 × 1478, but these scene images do not contain targets. Thus, we embed many target image chips of size 128 × 128 into the whole scene images, with one of the examples shown in Fig. 7(a). This operation is reasonable because both the target chips and the scene images are acquired by the same SAR sensor with a resolution of 0.3 m.
- *Target Detection:* Target detection is formulated as a binary classification problem and thus also solved using a similar A-ConvNets approach. This is a binary classification problem being target or clutter. The A-ConvNets architecture for target detection contains four convolution layers and three max pooling layers. Max pooling layers follow the first three convolution layers, using a pooling size of 2 × 2 with a stride of 2 pixels. The convolution stride is fixed to 1 pixel. No spatial zero padding is used in the convolution layer. From the first to the fourth convolution layers, the convolution filter size is 5 × 5, 5 × 5, 6 × 6, and 4 × 4, respectively. Each of the first three convolution layers has 32 feature maps. The fourth convolution layer has two output units, representing the probability of being classified as target or clutter.

The training data include target image chips and background clutter chips. The target chips are from the same set of images as before, whereas the clutter chips are randomly sampled from many different scene images, including meadow, trees, crops, and small buildings. Overall, in this work, both the training set and the validation set contain 2000 target images and 4000 clutter images. The trained binary classifier A-ConvNets is then used as a target detector, which takes sliding window input from the SAR images with embedded target image chips. The output of this step is a probability map of target presence, as shown in Fig. 7(b). Then, it is binarized with a threshold of 0.9, and subsequently, we compute the centroid of each connected region to indicate the position of each potential target, as shown in Fig. 7(c).

- *Target Recognition:* The detected centroids correspond to image chips which can be isolated out from the original SAR image. A bounding box of size 88 × 88 can be

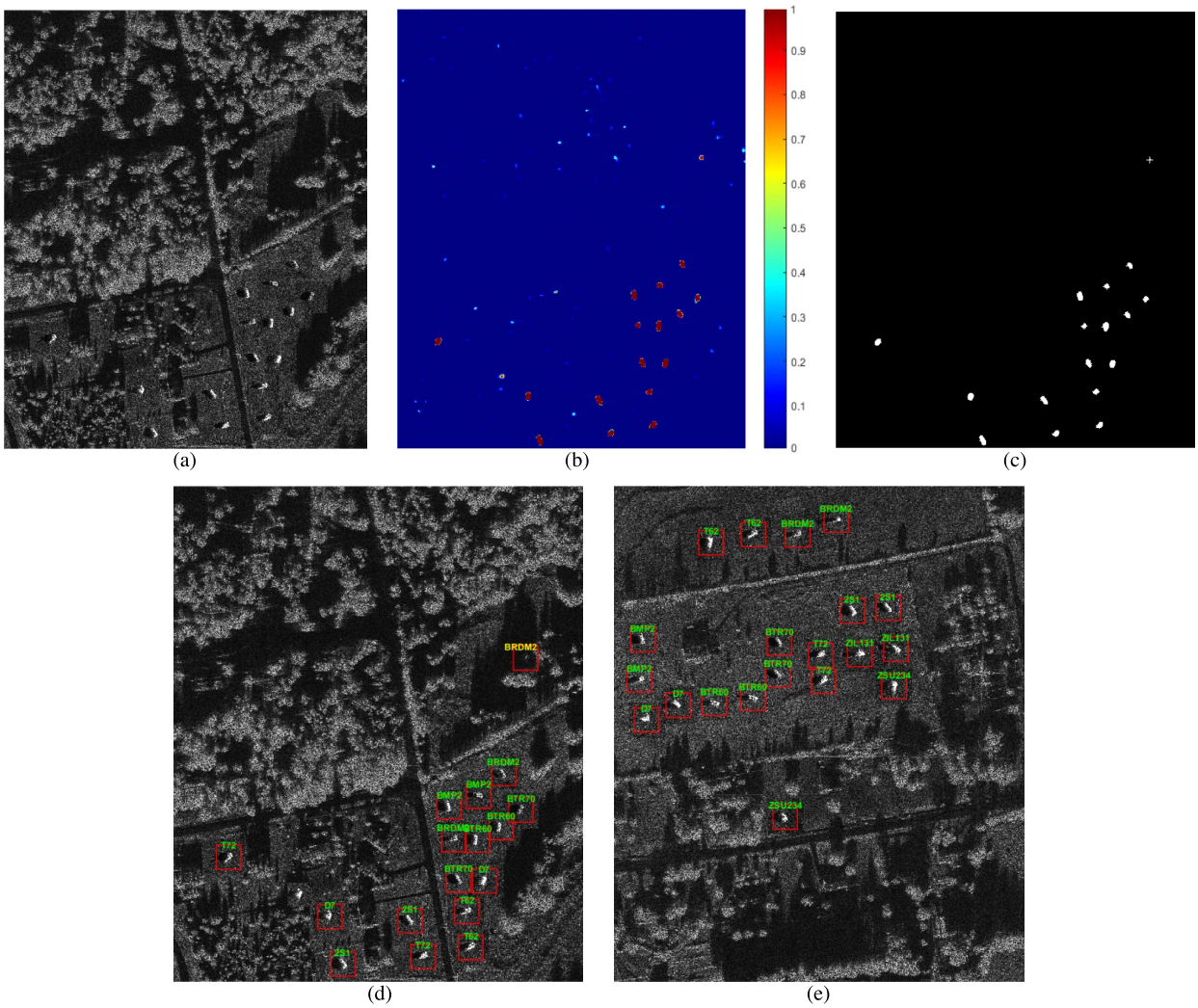

Fig. 7. (a) Whole scene SAR image with targets embedded. (b) Probability map of target presence. (c) Binarized probability map and the extracted centroid for each connected region. (d) Final target recognition results, with red boxes indicating the isolated image chips and letters indicating the recognized target types (green letter: correct; yellow letter: false alarm). (e) Another example of the final result.

labeled at their positions. Finally, these isolated chips are fed into the A-ConvNets trained for ten-target classification tasks. The final result is shown in Fig. 7(d), where green letters denote correct recognized target types, red letters denote false recognition (not existing in these two examples), and yellow letters denote false alarms. Another example is shown in Fig. 7(e).

V. Conclusion

Feature extraction plays a key role in the classification performance of SAR-ATR. Inspired by the recent great success of deep ConvNets in computer vision and speech recognition, we address the problem of feature extraction by constructing a multistage network to automatically learn hierarchical features from large data sets, instead of relying on hand-designed features. Whenever ConvNets was directly applied to SAR-ATR, the overfitting problem due to limited training data sets generated poor classification accuracy. A novel A-ConvNets was developed by removing the fully connected architectures in the traditional ConvNets, to reduce the number of independent trainable parameters. The network architecture, training details, and common rules for setting hyperparameters were also described in this paper. Experimental results on the benchmark MSTAR data set illustrate the effectiveness of A-ConvNets: a 99% accuracy of ten-class classification was achieved. Extensive studies on the robustness and confuser rejection were also conducted. The results show that our approach performs well under EOC and that a reasonable balance can be achieved between detection and false alarm ratios.

Acknowledgment

The authors would like to thank the anonymous reviewers for the constructive suggestions.

References

[1] D. E. Dudgeon and R. T. Lacoss, "An overview of automatic target recognition," *Lincoln Lab. J.*, vol. 6, no. 1, pp. 3–10, 1993.

[2] Y. Cui, G. Zhou, J. Yang, and Y. Yamaguchi, "On the iterative censoring for target detection in SAR image," *IEEE Geosci. Remote Sens. Lett.*, vol. 8, no. 4, pp. 641–645, Jul. 2011.

[3] J. I. Park and K. T. Kim, "Modified polar mapping classifier for SAR automatic target recognition," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 50, no. 2, pp. 1092–1107, Apr. 2014.
[4] L. M. Novak, G. J. Owirka, W. S. Brower, and A. L. Weaver, "The automatic target-recognition system in SAIP," *Lincoln Lab. J.*, vol. 10, no. 2, pp. 187–201, 1997.
[5] L. M. Novak, "State-of-the-art of SAR automatic target recognition," in *Proc. IEEE Int. Radar Conf.*, 2000, pp. 836–843.
[6] T. D. Ross, J. J. Bradley, L. J. Hudson, and M. P. O'Connor, "SAR ATR: So what's the problem? An MSTAR perspective," in *Proc. 6th SPIE Conf. Algorithms SAR Imagery*, 1999, vol. 3721, pp. 566–579.
[7] E. R. Keydel, S. W. Lee, and J. T. Moore, "MSTAR extended operating conditions: A tutorial," in *Proc. 3rd SPIE Conf. Algorithms SAR Imagery*, 1996, vol. 2757, pp. 228–242.
[8] A. Hirose, Ed., *Complex-Valued Neural Networks: Advances and Applications*. Hoboken, NJ, USA: Wiley-IEEE Press, 2013.
[9] Q. Zhao and J. C. Principe, "Support vector machines for SAR automatic target recognition," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 37, no. 2, pp. 643–654, Apr. 2001.
[10] Y. J. Sun, Z. P. Liu, S. Todorovic, and J. Li, "Adaptive boosting for SAR automatic target recognition," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 43, no. 1, pp. 112–125, Jan. 2007.
[11] C. F. Olson and D. P. Huttenlocher, "Automatic target recognition by matching oriented edge pixels," *IEEE Trans. Image Process.*, vol. 6, no. 1, pp. 103–113, Jan. 1997.
[12] N. M. Sandirasegaram, "Spot SAR ATR using wavelet features and neural network classifier," Def. R&D Canada, Ottawa, ON, Canada, DRDC Ottawa TM 2005-154, Tech. Memorandum, 2005.
[13] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.
[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.
[15] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[16] D. Ciregan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 3642–3649.
[17] Y. Sun, X. Wang, and X. Tang, "Deep learning face representation from predicting 10,000 classes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2014, pp. 1891–1898.
[18] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Proc. IEEE Eur. Conf. Comput. Vis.*, 2014, pp. 818–833.
[19] N. Srivastava *et al.*, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.
[20] O. Abdel-Hamid *et al.* "Convolutional neural networks for speech recognition," *IEEE/ACM Trans. Audio, Speech Language Process.*, vol. 22, no. 10, pp. 1533–1545, Oct. 2014.
[21] Y. LeCun, K. Kavukcuoglu, and C. Farabet, "Convolutional networks and applications in vision," in *Proc. IEEE Int. Symp. Circuits Syst.*, 2010, pp. 253–256.
[22] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier networks," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2011, pp. 315–323.
[23] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York, NY, USA: Springer-Verlag, 2006.
[24] J. Bouvrie, "Notes on convolutional neural networks," Center Biol. Comput. Learn., Massachusetts Inst. Technol, Cambridge, MA, USA, MIT CBCL Tech Rep., pp. 38–44, 2006.
[25] Y. Jia *et al.*, "Caffe: Convolutional architecture for fast feature embedding," in *Proc. ACM Int. Conf. Multimedia*, 2014, pp. 675–678.
[26] Y. Bengio, "Practical recommendations for gradient-based training of deep architectures," in *Neural Networks: Tricks of the Trade*. New York, NY, USA: Springer-Verlag, 2012, pp. 437–478.
[27] Y. LeCun *et al.*, "Efficient backprop," in *Neural networks: Tricks of the Trade*. Berlin, Germany: Springer-Verlag, 2012, pp. 9–48.
[28] The Air Force Moving and Stationary Target Recognition Database. [Online]. Available: https://www.sdms.afrl.af.mil/datasets/mstar/
[29] R. Singh and B. V. K. V. Kumar, "Performance of the extended maximum average correlation height (EMACH) filter and the polynomial distance classifier correlation filter (PDCCF) for multi-class SAR detection and classification," in *Proc. 9th SPIE Conf. Algorithms SAR Imagery*, 2002, vol. 4727, pp. 265–276.
[30] J. A. O'Sullivan, M. D. DeVore, V. Kedia, and M. I. Miller, "SAR ATR performance using a conditionally Gaussian model," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 37, no. 1, pp. 91–108, Jan. 2001.
[31] U. Srinivas, V. Monga, and R. G. Raj, "SAR automatic target recognition using discriminative graphical models," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 50, pp. 591–606, Jan. 2014.
[32] G. Dong, N. Wang, and G. Kuang, "Sparse representation of monogenic signal: With application to target recognition in SAR images," *IEEE Signal Process. Lett.*, vol. 21, no. 8, pp. 952–956, Aug. 2014.
[33] G. Dong and G. Kuang, "Classification on the monogenic scale space: Application to target recognition in SAR image," *IEEE Trans. Image Process.*, vol. 24, no. 8, pp. 2527–2539, Aug. 2015.
[34] K. Fukushima, "Neocognitron: A self organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biol. Cybern.*, vol. 36, no. 4, pp. 193–202, 1980.
[35] Q. Zhao *et al.*, "Synthetic aperture radar automatic target recognition with three strategies of learning and representation," *Opt. Eng.*, vol. 39, no. 5, pp. 1230–1244, 2000.
[36] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in *Proc. Int. Conf. Comput. Vis.*, 2015, pp. 1026–1034.

Sizhe Chen (S'14) received the B.E. degree in electronic engineering from Beihang University, Beijing, China, in 2013. He is currently working toward the M.S. degree in the Key Laboratory of Information Science of Electromagnetic Waves, Fudan University, Shanghai, China.

His research focuses on the application of machine learning and deep learning algorithms to synthetic aperture radar.

Haipeng Wang (S'03–M'06) received the B.S. and M.S. degrees in mechanical and electronic engineering from the Harbin Institute of Technology, Harbin, China, in 2001 and 2003, respectively, and the Ph.D. degree in environmental systems engineering from the Kochi University of Technology, Kochi, Japan, in 2006.

He was a Visiting Researcher with Waseda University, Tokyo, Japan, in 2008. He is currently an Associate Professor with the Key Laboratory of Information Science of Electromagnetic Waves (Ministry of Education), Department of Communication Science and Engineering, School of Information Science and Engineering, Fudan University, Shanghai, China. His research interests include signal processing, SAR image processing and analysis, speckle statistics, applications to forestry and oceanography, and machine learning and its applications to SAR images.

Dr. Wang has been a member of the Technical Program Committee of the IEEE International Geoscience and Remote Sensing Symposium since 2011. He was a recipient of the Dean Prize of the School of Information Science and Engineering, Fudan University in 2009.

Feng Xu (S'06–M'08–SM'14) received the B.E.(Hons.) degree in information engineering from Southeast University, Nanjing, China, in 2003 and the Ph.D.(Hons.) degree in electronic engineering from Fudan University, Shanghai, China, in 2008.

From 2008 to 2010, he was a Postdoctoral Fellow with the NOAA Center for Satellite Applications and Research (STAR), Camp Springs, MD, USA. From 2010 to 2013, he was with Intelligent Automation, Inc., Rockville, MD, while partly working for the NASA Goddard Space Flight Center, Greenbelt, MD, as a Research Scientist. In 2012, he was selected into China's Global Experts Recruitment Program, and subsequently, in June 2013, he returned to Fudan University, where he currently serves as a Professor in the School of Information Science and Technology and the Vice Director of the Ministry of Education Key Laboratory of Information Science of Electromagnetic Waves. He has authored or coauthored over 24 papers in peer-reviewed journals, two books, and two patents, among numerous conference papers. His current research interests include fast electromagnetic modeling for complicated target and environments, target reconstruction from multidimensional high-resolution SAR images, inverse scattering tomography, and SAR remote sensing applications in Earth observation.

Dr. Xu is the Founding Chair of the IEEE Geoscience and Remote Sensing Society Shanghai Chapter. He currently serves as an Associate Editor of the IEEE GEOSCIENCE AND REMOTE SENSING LETTERS. Among other honors, he was a recipient of the SUMMA graduate fellowships in the advanced electromagnetics area in 2007, the second-class National Natural Science Award of China in 2011, and the Early Career Award of the IEEE Geoscience and Remote Sensing Society in 2014.

Ya-Qiu Jin (SM'89–F'04) received the Bachelor's degree from Peking University, Beijing, China, in 1970 and the M.S., E.E., and Ph.D. degrees from Massachusetts Institute of Technology, Cambridge, MA, USA, in 1982, 1983, and 1985, respectively, all in electrical engineering and computer science.

He was a Research Scientist with the Atmospheric Environmental Research (AER), Inc., Cambridge, in 1985, a Research Associate Fellow with the City University of New York, New York, NY, USA, in 1986–1987, and a Visiting Professor with the University of York, York, U.K., in 1993, sponsored by the U.K. Royal Society. He is currently the Te-Pin Professor and the Director of the Key Laboratory of Information Science of Electromagnetic Waves [Ministry of Education (MoE)] with Fudan University, Shanghai, China. He has authored or coauthored over 720 papers in refereed journals and conference proceedings and 14 books, including *Polarimetric Scattering and SAR Information Retrieval* (Wiley and IEEE, 2013), *Theory and Approach of Information Retrievals from Electromagnetic Scattering and Remote Sensing* (Springer, 2005), and *Electromagnetic Scattering Modelling for Quantitative Remote Sensing* (World Scientific, 1994). His main research interests include electromagnetic scattering and radiative transfer in complex natural media, microwave satellite-borne remote sensing, as well as theoretical modeling, information retrieval and applications in Earth terrain and planetary surfaces, and computational electromagnetics.

Dr. Jin is the Academician of the Chinese Academy of Sciences and a Fellow of The World Academy of Sciences (the Developing Countries Academy of Sciences), the International Academy of Astronautics, and The Electromagnetics Academy. He is an IEEE Geoscience and Remote Sensing Society (GRSS) Distinguished Speaker and an Associate Editor of IEEE ACCESS. He was a member of the IEEE GRSS Administrative Committee, a Chair of the IEEE Fellow Evaluation of GRSS (2009–2011), and a Cochair of the Technical Program Committee for the International Geoscience and Remote Sensing Symposium (IGARSS) 2011 in Vancouver, Canada. He will be a Cogeneral Chair for IGARSS 2016 in Beijing. He was an Associate Editor of the IEEE TRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING (2005–2012). He was a recipient of the Senior Research Associateship in the National Oceanic and Atmospheric Administration/National Environmental Satellite, Data, and Information Service by the U.S. National Research Council in 1996. He was also a recipient of the first-grade MoE Science Prizes (1992, 1996, and 2009), the first-grade Guanghua Science Prize (1993), the IEEE GRSS Education Award (2010), the China National Science Prize (1993, 2011), the Prize of Fudan President (2004, among many other prizes), the Shanghai Sci/Tech Gong Cheng Award (2015), and the IEEE GRSS Distinguished Achievement Award (2015).