Received May 12, 2019, accepted June 3, 2019, date of publication June 13, 2019, date of current version June 28, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2922844

Adversarial Deep Domain Adaptation for Multi-Band SAR Images Classification

WEI ZHANG, YONGFENG ZHU, AND QIANG FU

National Key Laboratory of Science and Technology on ATR, Institution of Electronic Science, National University of Defense Technology, Changsha 410073, China

Corresponding author: Yongfeng Zhu (zhuyongfeng@nudt.edu.cn)

ABSTRACT Deep convolutional neural networks (CNNs) have made a breakthrough on supervised SAR images classification. However, SAR imaging is considerably affected by the frequency band. That means a neural network trained on a SAR image set of one band is not suitable for the classification of another band images. As manually labeling the training samples of each band is always time-consuming, we propose an unsupervised multi-level domain adaptation method based on adversarial learning to solve the problem of multi-band SAR images classification. First, we train a discriminative CNN using samples of one frequency band data set that contains labels to map the data to a latent feature space. Then, we adjust the trained CNN to map the unlabeled samples of another frequency band data set to the same feature space through alternately optimizing two adversarial loss functions. Thus, the features of these two band images are fused and can be classified by the same classifier. We checked the performance of our method using both simulated data and measured data. Our method made a breakthrough in the classification of multi-band images with accuracies of 99% on both data sets. The results are even very close to the supervised CNN trained using a large number of labeled samples.

INDEX TERMS Convolutional neural network(CNN), domain adaptation, multi-band SAR images classification, adversarial learning.

I. INTRODUCTION

Synthetic aperture radar (SAR) is one of the important measures for earth observation, due to its advantages of all-day, all-weather, and high resolution. SAR images contain rich material, geometry and structure information about observed targets and scenes. Thus, SAR image classification has become a hot topic in academia. Traditional methods including artificial neural network [1], support vector machine (SVM) [2], [3], Markov random field (MRF) [4], [5], AdaBoost [6], and so on [7]–[10] made early contributions in the area of automatic SAR images classification. But they all rely on expert knowledge to extract valid features from images. In the past few years, deep learning methods such as convolution neural network (CNN) have made breakthroughs in computer vision [11]–[13] and rapidly expanded to other fields, including SAR image classification. After being trained in a large data set, deep CNN always achieves higher performance in SAR image classification than traditional method [14]–[17].

The associate editor coordinating the review of this manuscript and approving it for publication was Aysegul Ucar.

Despite its powerful in image classification, CNN as a data-hungry method requires a lot of labeled training samples to avoid overfitting. But there's still no sufficient annotated SAR data set, as data annotation is a time-consuming work and needs domain knowledge. In addition, deep neural networks are also well known for their weak generalization. One pixel change can fool the deep neural network [18]. Since SAR images of different frequency bands are very different, a CNN trained on samples of one frequency band may fail completely when classifying images of another band. Some researches have been done to reduce the dependence on data annotations, [19] and [20] utilized the pre-trained CNN to learn transferable knowledge of image classification and then fine-tuned the model on small numbers of labeled SAR samples. These methods boost the performance of CNNs trained on small training sets, but are still supervision-required. [21] took advantage of zero shot learning and deep generative network to produce semantic feature representation of SAR images. This method doesn't need any supervised information but is only used for feature representation not for classification. For several types of concerned targets or scenes, how

to adapt a CNN trained in the data set of one frequency band to the classification of another band samples with absolutely unsupervised settings remains a problem.

In this paper, we introduce domain adaptation to address the problem. Domain adaptation is an area related to machine learning and transfer learning, which aims to solve the problem of data set bias. There are two main concepts in domain adaptation, which are source domain and target domain. Source domain denotes data sets whose annotations are available but are different from the test set. Target domain is similar to the test set but usually lacks labels. The main idea of domain adaptation is using the source domain train a supervised classifier, as well as adjusting the model in an unsupervised way to learn domain-invariant features from both of the source domain and the target domain. In deep learning regime, the combination of domain adaptation and deep learning enhances the generalization of deep neural networks. Some researches have been done in the area of computer vision, [22]–[24] utilized some mathematical metrics such as Maximum Mean Discrepancy(MMD) or Joint Maximum Mean Discrepancy(J-MMD) to measure the distance of feature distributions between the source domain and the target domain. Then, they minimize these metrics to fuse the source domain and the target domain in feature level. These methods work well in certain data sets. However, these metrics-based methods usually fail when applied to the classification of multi-band SAR images due to the considerable variations of SAR images in different frequency bands. Meanwhile, generative adversarial networks(GAN) [25] emerged as a powerful framework for producing high fidelity data due to its adversarial learning mechanism. The GAN framework learns a discriminator to distinguish between real and generated samples as accurately as possible, as well as a generator which aims to ''fool'' the discriminator by creating samples as close to real data as possible. Due to the superiority of adversarial learning, some studies [26], [27] have introduced this training method to domain adaptation. These methods usually achieve good capability in the classification of optical images. But they only adjust the end layer of the neural network, which may be restrictive in that the hidden features of several previous layers are not transferable. Thus, these methods are usually not effective for SAR images that vary greatly with frequency bands.

We propose a adversarial domain adaptation method that take advantage of multi-level features(MLADA) to achieve multi-band SAR image classification, illustrated in Figure 1. We first train a CNN using the labeled source data to learn discriminative feature representation. Then, a separate target encoder is learned through two domain adversarial losses to map the target data to the same feature space as the source data. The main works of this paper are summarized as follows.

- The paper makes the first attempt on adversarial deep domain adaption to solve the multi-band SAR images classification problem. For each component of our method, we design a robust encoder using the residual block [13] to integrate multi-level features and a multi-layer discriminator that shares feature maps with the encoder to distinguish the multi-level features of the source domain and target domain.
- Three data sets are utilized to test the performance of MLADA. Firstly, a simulated data set of six-category targets on X-band and C-band is adopt to evaluate the performance of SAR target recognition. Secondly, a set of real images of five different scenes on X-band and C-band are utilized to test the performance in the real scenario. Thirdly, we test the model's transferability between a optical data set and a SAR data set. MLADA achieves 99% accuracy in both simulated data and measured data and 62% accuracy in the transfer test between optical and SAR images. The results prove that MLADA effectively improves the extensiveness of the deep CNN in SAR image classification. We also check the anti-noise performance of our method using noisy simulated data and find that our method is reliable as long as the SNR is greater than 15db.
- We utilize a gradient-based visualization approach [28], [29] to do some reliability analysis about MLADA. And the results present that the domain

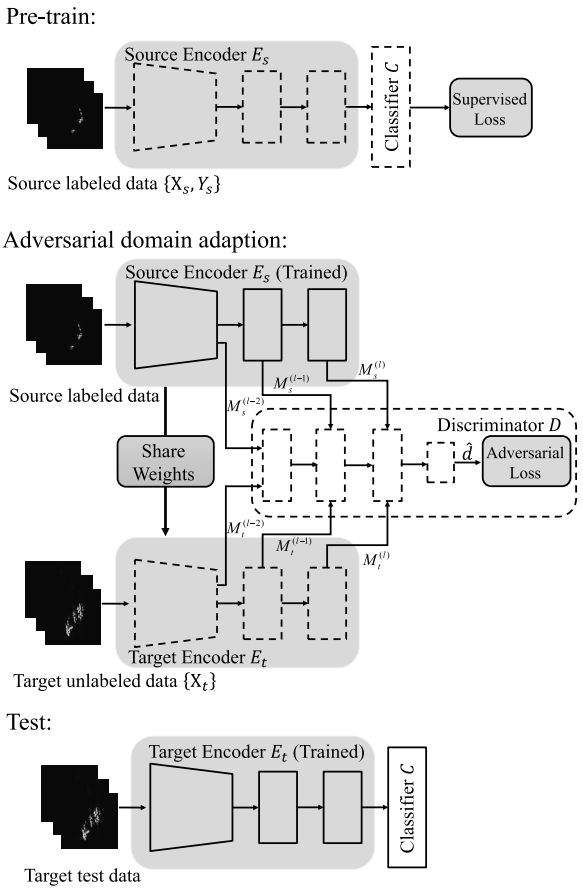

FIGURE 1. Flow chart of the algorithm implementation, where the dotted boxes represent the components need to be trained. The target encoder is initialized by the trained source encoder.

adaptation model can roughly highlight the pixels of interest which contains discriminative information even at low SNR conditions, which is similar to the supervised CNN directly trained in the target domain. However, the model trained only in the source domain always misses the points.

The rest of the paper will be arranged as follows. In section II, the overall architecture and the detail configurations of the method will be illustrated. In section III the experiment settings will be introduced. In section IV the results and discussions will be demonstrated. And finally, in section V we give the conclusion.

II. METHODOLOGY

Here, we first introduce the overall architecture of our method and then describe the detail configurations of each components. Finally, we demonstrate the specific implementation of training the model.

A. OVERALL ARCHITECTURE

Figure 1 shows the flow chart of the whole method, where the solid boxes represent fix components of the model during training, while the dotted boxes represent the components that need to be trained. In domain adaption, we assume that the source domain images with their labels $\{X_s, Y_s\}$ and the target domain images without labels $\{X_t\}$ are accessible, where the target domain is considered to contain the same classes as the source domain. We use $M_t^{(l)}$ and $M_s^{(l)}$ to represent the output feature maps of the l-layer target encoder E_t and source encoder E_s, respectively.

The final objective of our method is to learn a $E_t : X_t \rightarrow M_t^{(l)}$ and a target classifier $C_t : M_t^{(l)} \rightarrow R^N$ to classify the target domain images into correct one of the N categories. However, the assumption is that the target domain contains no supervised information. Thus, we can not train E_t and C_t directly using the target domain samples. Considering that the source domain's labels Y_s are available, we first train a supervised source encoder $E_s : X_s \rightarrow M_s^{(l)}$, as well as a source classifier $C_s : M_s^{(l)} \rightarrow R^N$, instead. And then adjust the trained model to be reused in the target domain through domain adaption.

The main idea of domain adaptation is regularizing the training of source encoder E_s and target encoder E_t to minimize the distribution distance between their feature maps $E_s(X_s)$ and $E_t(X_t)$. Thus, both E_s and E_t will extract similar domain-invariant features. In that case, the source classifier C_s can be directly reused to classify the target feature representation without any adjustments. Hence, we use a unified classifier C instead, that is $C = C_t = C_s$.

C outputs an N-dimensional vector that represents the confidence probability of each category computed by softmax function [30]

$$C_i = \frac{exp(c_i')}{\sum_{j=1}^{N} exp(c_j')}, \tag{1}$$

where c_j' denotes the jth element of the vector input to the function. Then, the source encoder E_s and classifier C can be trained in the source domain using the supervised loss:

$$\min_{E_s,C} L_{cls}(x_s, y_s) = -\mathbb{E}_{\{x_s,y_s\}\sim p\{X_s,Y_s\}} \sum_{i=1}^{N} \mathbb{1}_{[i=y_s]} log[C(E_s(x_s))_i]. \tag{2}$$

Now, we are able to implement domain adaptation through adversarial learning. On the one hand, we train a domain discriminator $D : \{M_{s,t}^{(l)}, M_{s,t}^{(l-1)}, M_{s,t}^{(l-2)}\} \rightarrow R$ to distinguish the features of the source domain and the target domain. As illustrated in Figure 1, D shares the three-level feature maps, $\{M_{s,t}^{(l)}, M_{s,t}^{(l-1)}, M_{s,t}^{(l-2)}\}$, with the source encoder E_s and the target encoder E_t, and produce a scalar d indicating the domain class. In practice, d is normalized by the Sigmoid function:

$$\hat{d} = \frac{1}{1 - exp(-d)}. \tag{3}$$

We consider $\hat{d}$ as the probability that the input sample x belongs to the source domain. In other words, for a source domain sample x_s, we expect that $\hat{d} = D(E_s(x_s))$ tends to 1. While for a target domain sample x_t, we expect that $\hat{d} = D(E_t(x_t))$ tends to 0. Thus, we can train D with the following adversarial loss function:

$$\min_{D} L_{adv_D}(x_s, x_t) = -\mathbb{E}_{x_s\sim pX_s}[logD(E_s(x_s))] - \mathbb{E}_{x_t\sim pX_t} log[1 - D(E_t(x_t))]. \tag{4}$$

On the other hand, the target encoder E_t is initialized using the trained parameters of E_s. And then, E_t learns to confuse D by mapping the target data x_t to the same feature space as source feature maps $E_s(x_s)$. That is, for a target domain sample x_t, we expect to make $\hat{d} = D(E_t(x_t))$ tend to 1 by training E_t. Thus, the adversarial loss of E_t is given as follow:

$$\min_{E_t} L_{adv_{E_t}}(x_s, x_t) = -\mathbb{E}_{x_t\sim pX_t} log[D(E_t(x_t))]. \tag{5}$$

Formulas 4 and 5 are trained alternately and balance each other. In theory, we will finally get a robust encoder E_t that maps x_t into the similar feature space to $E_s(x_s)$.

B. DETAIL CONFIGURATIONS OF EACH COMPONENT

Figure 2 illustrates the framework of each component and the connections between them. In this paper, we build the encoder as a very deep convolution neural network using the well-known residual block [13]. Each convolutional layer maps the outputs of the previous layer to higher-level feature maps through the forward propagation function

$$O_j^{(l)}(w, h) = \sigma[\sum_{i=1}^{N} \sum_{u,v=0}^{K-1} k_{ij}^{(l)}(u, v) O_i^{(l-1)}(w - u, h - v)], \tag{6}$$

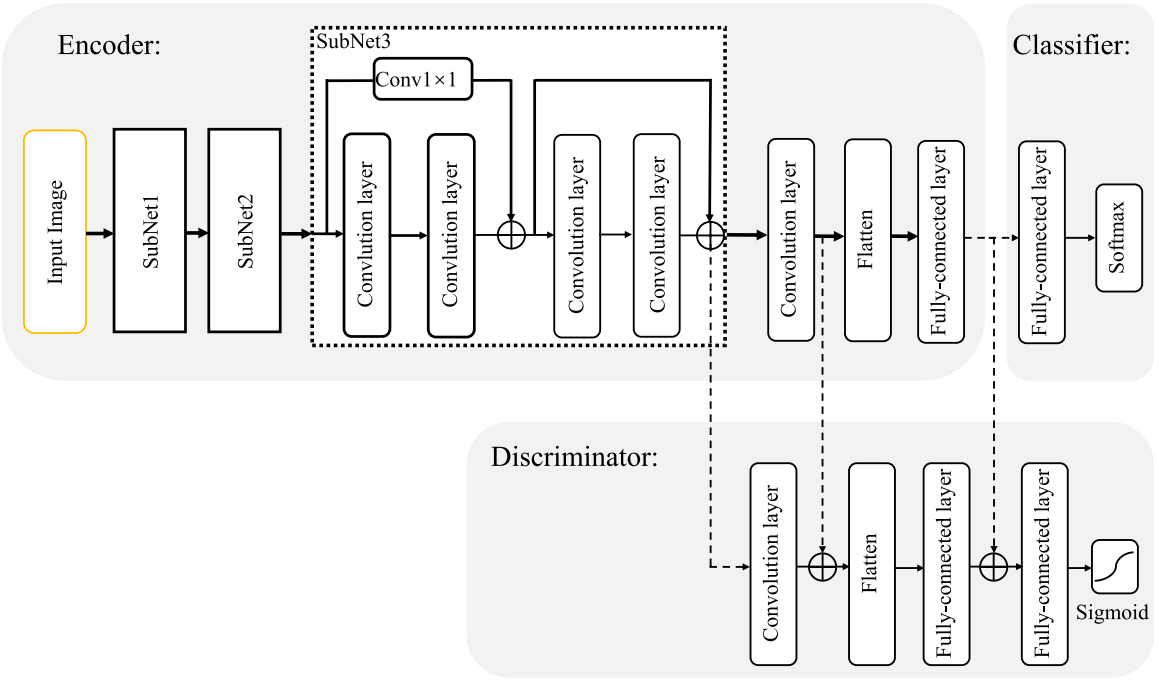

FIGURE 2. Detail configurations and connections of each component.

where $O_j^{(l)}(w, h)$ represents the jth output feature maps of the lth convolutional layer at the position (w, h). N denotes the number of channels of the input feature maps. And $k_{ij}^{(l)}(u, v)$ denotes the trainable convolution kernel with the size of $K \times K$. $\sigma(\cdot)$ is the nonlinear activation function. In this paper, the Rectified Linear Unit(ReLU) is used as the activation function, which is given by

$$\sigma(x) = max(0, x). \tag{7}$$

Consider a $W_1 \times W_1$ feature map as the input of a convolutional layer, the convolutional layer can produce several new feature maps of size $W_2 \times W_2$. The scale of output feature maps can be computed by $W_2 = (W_1 - K + 2P)/S + 1$, where S denotes the stride of kernels, and P is the padding of the input feature map.

To be modularized, we package every two modules in a group into a subnetwork, as illustrated in SubNet3. As we can see, each residuals block contains two convolutional layers and one shortcut bypass that directly transfer the previous feature maps to the high layers. Lots of experimental results indicate that this architecture is powerful in performance and does not consume too much memory. The shortcut bypass of the first residual block is a 1×1 convolutional layer without non-linear transformation. It only regularizes the input features to the same scale and number of channels as the output features. We express the two convolutional layers of the first residual block as the non-linear transformation $H_1(\cdot)$, and $H_2(\cdot)$ for the second. The 1×1 convolutional layer is essentially a linear transformation, which is represented by $L(\cdot)$. Hence, consider the feature maps $O^{(l)}$ of the lth layer to be fed into the subnetwork, the forward transition of the subnetwork is shown as follows

$$O^{(l+2)} = H_1(O^{(l)}) + L(O^{(l)}), \tag{8}$$
$$O^{(l+4)} = H_2(O^{(l+2)}) + O^{(l+2)}. \tag{9}$$

[12] proposed that for a deep CNN, the stack of multiple small convolution kernels(e.g., 1×1 or 3×3) can achieve similar performance to one large kernel(e.g., 7×7 or 9×9) but has only half the parameters. In addition, deeper neural networks can include more nonlinear transformations, which always means stronger learning ability. Thus, we build the network using the small kernels of size 3×3. The number of subnetworks is set to three by the limitation of memory. The last convolutional layer is designed to reduce the number of feature maps to be passed to the fully connected layer to avoid producing so many parameters that may overflow the memory. The detail layer configurations are listed in TABLE 1, where the convolutional layers are represented as "Conv.(number of feature maps)@(filter size and stride)."

A batch normalization [31] layer is added between each convolution kernel and the activation function. That's a deep neural network training trick, which can accelerate the convergence speed of the model by normalizing the values to be activated to the interval with a large derivative of the activation function. Consider a mini-batch contains m samples, while x_j is the kernel outputs of the jth sample. The specific implementation of batch normalization is shown as follows

$$\mu(c, w, h) = \frac{1}{m} \sum_{j=0}^{m} x_j(c, w, h), \tag{10}$$

TABLE 1. Detail layer configurations of the encoder.

Blocks	Layers	Shortcut
SubNet1	Conv.128@3 × 3 stride×2 Batch Normalization ReLU Conv.128@3 × 3 stride×1 Batch Normalization ReLU	Conv.128@1 × 1 stride×2 Batch Normalization
	Conv.128@3 × 3 stride×2 Batch Normalization ReLU Conv.128@3 × 3 stride×1 Batch Normalization ReLU	Identity mapping
SubNet2	Conv.256@3 × 3 stride×2 Batch Normalization ReLU Conv.256@3 × 3 stride×1 Batch Normalization ReLU	Conv.256@1 × 1 stride×2 Batch Normalization
	Conv.256@3 × 3 stride×2 Batch Normalization ReLU Conv.256@3 × 3 stride×1 Batch Normalization ReLU	Identity mapping
SubNet3	Conv.512@3 × 3 stride×2 Batch Normalization ReLU Conv.512@3 × 3 stride×1 Batch Normalization ReLU	Conv.512@1 × 1 stride×2 Batch Normalization
	Conv.512@3 × 3 stride×2 Batch Normalization ReLU Conv.512@3 × 3 stride×1 Batch Normalization ReLU	Identity mapping
Conv1	Conv.128@3 × 3 stride×1 Batch Normalization ReLU Flatten	None
Feature	Fc layer nodes×512 ReLU	None

$$\sigma^2(c,w,h) = \frac{1}{m}\sum_{j=0}^{m}[x_j(c,w,h) - \mu(x,w,h)]^2, \quad (11)$$

$$\hat{x}_j(c,w,h) = \frac{x_j(c,w,h) - \mu(c,w,h)}{\sqrt{\sigma^2(c,w,h) + \epsilon}}, \quad (12)$$

$$y_j(c,w,h) = \gamma\hat{x}_j(c,w,h) + \beta, \quad (13)$$

where c, w, h index the pixels in x_j, μ and σ denote the mean and standard deviation, respectively. ϵ is a tiny constant to keep the denominator from being zero, and is set to 1e-8 in this paper. It is generally considered that the feature maps of intermediate layers obey an approximate Gaussian distribution, $x \sim N(\mu, \sigma)$. Hence, formulas 10-12 normalize the distribution to an approximate standard normal distribution, $\hat{x} \sim N(0, 1)$. β and γ are trainable parameters which control the scale and offset of y_j. These two parameters are optimized by error back-propagation.

The discriminator D and the classifier C are two shallower neural networks comparing with the encoder E. Their configurations are recorded in TABLE 2. It should be noted that D shares three feature layers with E. The each layer's output of D is given as follow:

TABLE 2. Detail layer configurations of the classifier and discriminator.

Components	Layer Configurations
Classifier	Input layer @nodes×512 ReLU Category layer @nodes×C(C=number of cateegories) Softmax
Discriminator	Conv.128@3 × 3 stride× 1 Batch Normalization ReLU Flatten Hidden layer @nodes×512 ReLU Discriminative layer @nodes×1 Sigmoid

$$d^{(l)} = D_l[\alpha d^{(l-1)} + E_l(x)], \quad x \in \{X_s, X_t\}, \quad (14)$$

where l is the current layer, α is the decay factor of the forward transition which is set to 0.2 through cross validation in the paper. D_l and E_l are the forward propagation function of the current layer in D and E, respectively.

C. TRAINING DETAILS

As in most heuristic algorithm, all the trainable parameters in deep neural networks are adapted to minimize the loss function. Normally, it is impossible to calculate the global minimum of the loss function. Thus, deep learning takes advantages of some gradient-based optimizer to approach the optimal solution of the loss function iteratively. The most basic of these algorithms is gradient descent, which is optimized by the updating rules for parameters: $w \leftarrow w - \eta(\partial L/\partial w)$, where η is a hyper-parameter called learning rate. In multi-layer neural networks, the partial derivatives of loss with respect to trainable parameters on each layer can be calculated by the error back-propagation algorithm. However, there are some particular settings of the optimizer for each step of our training.

1) ADAPTIVE MOMENT ESTIMATION

The Stochastic Gradient Descent(SGD) [33] algorithm is widely used to optimize neural networks of various architectures. Empirically, the learning rate is set to decay every few iterations to keep the training process stable. Which iteration or how much to reduce the learning rate are all preset by users and generally cannot be adjusted in real time during training. As mentioned above, the target encoder E_t and the discriminator D are mutually influential in adversarial learning. Minimizing $L_{adv_{E_t}}$ will have the opposite impact on L_{adv_D}. Thus, if we train the target encoder and discriminator alternately, the change of $L_{adv_{E_t}}$ and L_{adv_D} will be unstable, which requires the optimizer to adjust itself adaptively during training. Adaptive Moment Estimation(Adam) [32] is an adaptive optimizer that can increase the stability of training. Instead of configuring each step of the training, Adam comprehensively considers the first and secondary moment

TABLE 3. Optimizer configurations.

Step	Configurations
Pre-train	$L = L_{cls}$ $\eta = 1e-4$ $\beta_1 = 0.99, \beta_2 = 0.999$ $\epsilon = 1e-8$ $w \in \{E_s, C\}$
Domain adaptation(D)	$L =$ $\eta = 1e-5$ $\beta_1 = 0.99, \beta_2 = 0.999$ $\epsilon = 1e-8$ $w \in \{D\}$
Domain adaptation(E_t)	$L = L_{adv_{E_t}}$ $\eta = 1e-5$ $\beta_1 = 0.99, \beta_2 = 0.999$ $\epsilon = 1e-8$ $w \in \{E_t\}$

estimates of the gradient to compute the learning rate for each iteration. Consider a loss function L, the update rule for trainable parameters w is

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)\left\langle \frac{\partial L}{\partial w_t} \right\rangle_t, \tag{15}$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2)\left\langle \frac{\partial L}{\partial w_t} \right\rangle_t^2, \tag{16}$$

$$\hat{m}_t = \frac{m_t}{1-\beta_1^2}, \tag{17}$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^2}, \tag{18}$$

$$w_t = w_{t-1} - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon}\hat{m}_t, \tag{19}$$

where η is the initial learning rate; hyper-parameters β_1 and β_2 are the decay rates of the first and secondary moments; subscript t is the number of iteration; m_t and v_t are the first moment and secondary moment calculated in the tth iteration. $\langle \partial L/\partial w_t \rangle_t$ is the gradient of loss function with respect to w_t, which is computed by the tth mini-batch. ϵ is a tiny constant that prevents the denominator from being zero. The configurations of the optimizer for each step of our algorithm are listed in TABLE 3. In practice, β_1, β_2 and ϵ are set as the recommended values that are given by the inventor of Adam optimizer. And experiments show that such parameter setting is suitable for training. The initial learning rates are set according to the variation of the losses during the training. We first set η of all of the three losses functions to an empirical value, $1e-4$. For the pre-training step, this learning rate works well with L_{cls} decaying rapidly and being relatively stable. While for L_{adv_D} and $L_{adv_{E_t}}$, this learning rate leads to violent fluctuations of the losses. Thus, we reduce the learning rate to stabilize the losses. After several rounds of experiments, η of L_{adv_D} and $L_{adv_{E_t}}$ is finally set to $1e-5$.

2) TRAINABLE PARAMETERS INITIALIZATION

For the first step, the source encoder and the classifier are initialized randomly. Recent studies suggest that the deep neural networks with ReLU activation function should be initialized by a zero-mean normal distribution. The standard deviation should be set to $\sqrt{2/D}$ [34], where D is the dimension of the previous layer. For the second step, the initialization of discriminator is the same as above. However, the parameters of target encoder are initialized by sharing parameters with the pre-trained source encoder.

3) EARLY STOPPING

According to the derivation in [25], when the model reaches its global optimal solution through adversarial learning, $D(E_t(X_t)) = D(E_s(X_s)) = 0.5$. Thus, we can judge whether the training converges to the global optimum according to the output of D. We compute the average output of D for each iteration during the training, denoted as $\mathbb{E}(D)$. When $\mathbb{E}(D)$ reaches 0.5 or stabilizes to values near it for a few iterations, the training should be stopped in advance to avoid the results going worth again. Sometimes $\mathbb{E}(D)$ cannot converge to 0.5 during the training, at this time we can adjust the training times of D and E_t to balance them. When $\mathbb{E}(D)$ is too large, train E_t for more times; when $\mathbb{E}(D)$ is too small, train D for more times.

III. THE EXPERIMENTAL SETUP

The experiments are set up to verify the effectiveness of the algorithm proposed in this paper. A simulation data set and a measured data set are utilized for test. Both of these two data sets contain two subsets, temporarily recorded as A and B, which are composed of two-band SAR images. We test our method on two directions: $A \rightarrow B$ and $B \rightarrow A$, where the arrow points from the source domain to the target domain. Only the source domain retain labels. Meanwhile, we compare the performance of our method with general supervised CNN trained in the source domain and the target domain (30% and 100% of the training samples) and some other domain adaptation methods. In addition, we also investigate the feasibility of transferring the learned information from the optical images to SAR images. The remainder of this section describes the data sets and some detail configurations used in the experiments.

A. SIMULATION DATA SET

Due to the lack of measured multi-frequency radar echo data for target recognition, we utilize the commercial electromagnetic simulation software (CST) to estimate the radar reflectivity of the target 3D-CAD model. We collect models of six classes (car, trailer, bus, truck, land roller and jeep) for the test. The geometry parameters of simulation–azimuth angle ϕ, pitch angle θ–are shown in Figure 3.

The range of object azimuth angle covers the full 360° with the step size $\Delta\phi = 0.1°$ for each desired pitch angle. Each of the adjacent 128 angles is equivalent to one aperture. The azimuth resolution is determined by $\lambda/2\sigma$ with λ and σ being the wavelength and azimuth angle interval used, respectively. In the experiment, we produced a frame of image every $2\Delta\phi$ of azimuth. Thus, for any pitch angle, we obtained 1800 simulation images for each class. Meanwhile, the simulation

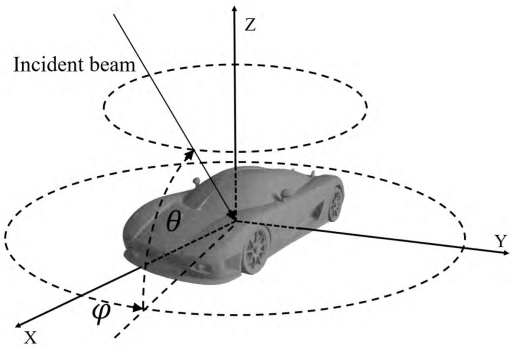

FIGURE 3. Coordinate system used for simulation.

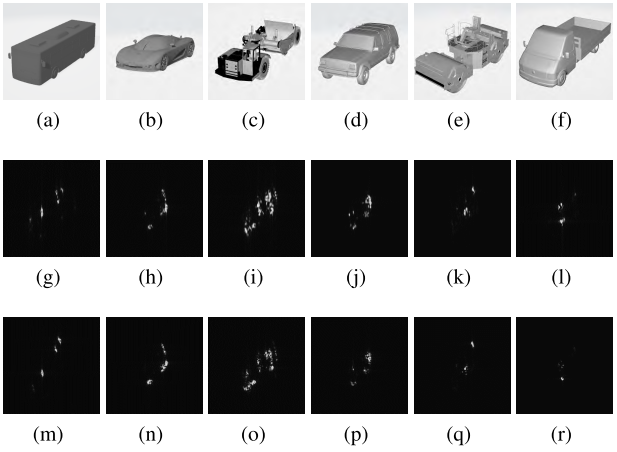

FIGURE 4. The simulation SAR samples of C-band and X-band. (g)-(l) represent the C-band images, (m)-(r) represent the X-band images.

wave bands are selected from C-band and X-band, which are 4.7-5.3Ghz and 9.7-10.3Ghz, respectively. The range resolution can be computed by $c/2B$, where c is the speed of light, and B is the frequency bandwidth. Thus, these two bands get the same range resolution. While the azimuth resolution of the X-band is approximately twice that of the C-band, that means that the images of the two bands will have some distortion between each other. Thus, we calibrate the images of the two bands to the same scale by bilinear interpolation. Figure 4 shows some samples of the simulation data.

For X-band, we set the pitch angles to 30° and 32°, while for c-band, we set them to 33° and 35°. Thus, each object model can produce 3,600 images in each band. By this means, we can not only test the performance of the algorithm in the multi-band target recognition but also check its effectiveness in multi-pitch angles.

To simulate the interference during electromagnetic wave transmission and verify the anti-noise performance of the algorithm, we add noise into the simulated echo for the data in the target domain, while the source domain is still composed of the perfect simulation data. The Signal Noise Ratio (SNR) covers 10db, 15db, 20db, 25db and 30db. Some samples of SAR images under each SNR are demonstrated in Figure 5.

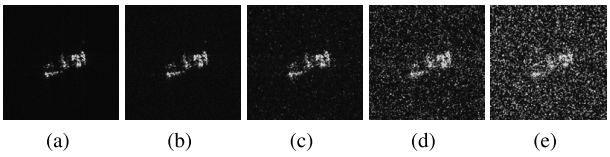

FIGURE 5. The simulation images under random noise with the SNR of (a) 30db, (b) 25db, (c) 20db, (d) 15db, (e) 10db.

FIGURE 6. Procedure of building the measured data set.

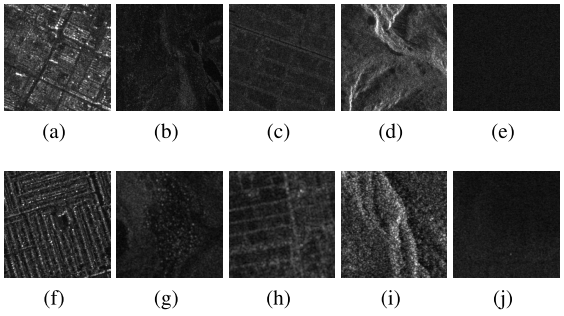

FIGURE 7. Samples of measured data, where (a)-(e) are obtained from Sentinel-1 and (f)-(j) are obtained from TerraSAR-X. From left to right are samples of cities, deserts, farmlands, mountains and oceans.

B. MEASURED DATA SET

We also check the effectiveness of our method on the classification of measured SAR scene images provided by Sentinel-1 and TerraSAR-X, two earth-observation satellites operating in the C-band and X-band, respectively. The strip map SAR images of 5 × 5m resolution obtained from Sentinel-1 and the more precise data of 1 × 1m resolution obtained from TerraSAR-X are collected for the experiment. All the SAR data sets obtained from the websites of Sentinel-1 and TerraSAR-X provide the packages that contain the region locations of the SAR images. We can view the optical remote sensing images of the same regions as the downloaded SAR images through the Google Earth. Through observing the optical images, we can label the categories of different scenes of the corresponding areas in the SAR images according to the landforms.

We labeled five main classes of scenes in the obtained SAR images, including cities, deserts, farmlands, mountains, and oceans, and select these scenes using rectangle bounding boxes. The bounding boxes should avoid crossing the boundaries of different scenes to ensure that each of them only contains one single class of scene. Then we cropped the selected scenes into 192 × 192 pixels with overlap of 50 pixels for each adjacent two slices within the bounding boxes. Figure 6 illustrate the procedure of building the

TABLE 4. The experiment results of the adversarial domain adaptation and some comparisons, where SimuC→SimuX and SimuX→SimuC denote that the experiments are performed the simulation data set; MeasC→MeasX and MeasX→MeasC represent the experiments on the measured data set; while Optical→SAR checks the transferability between SAR and optical images.

Method	SimuC→SimuX	SimuX→SimuC	MeasC→MeasX	MeasX→MeasC	Optical→SAR
CNN(Source only **baseline**)	0.9631±0.0024	0.9713±0.0032	0.3026±0.021	0.4305±0.0008	0.2156±0.0024
DDC	0.9282±0.008	0.9312±0.018	0.5719±0.0042	0.8396±0.004	0.4±0.0
DAN	0.9743±0.008	0.9596±0.012	0.6816±0.0032	0.9063±0.0064	0.4±0.0
ADDA	0.9856±0.004	0.9843±0.008	0.6787±0.0084	0.8973±0.0032	0.4167±0.0016
CNN(target 30% training samples)	0.6623±0.0146	0.7128±0.008	0.9664±0.0032	0.9823 ±0.0134	0.9735±0.0065
CNN(target 100% training samples)	0.9989±0.0001	1.0±0.0	0.9994±0.0024	1.0±0.0	0.9985±0.0012
MLADA(Our)	0.9920±0.0018	0.9992±0.0048	0.9835±0.0112	0.9971±0.0022	0.6239±0.0228

measured data set. Each data set contains 18,200 slices of SAR scene images, about 3,500 for each category. Figure 7 provides some samples of the cropped images.

C. DATA SET USED FOR TRANSFER BETWEEN OPTICAL IMAGES AND SAR IMAGES

This part is performed on a TerraSAR-X image and optical satellite image of the same scale. The SAR image is produced by a region of Visakhapatnam, India with VV polarization and resolution of 0.3 × 0.3m. While the optical satellite image is captured from the same area as that of the SAR image in Google Earth. The measure that we label the scenes and build the data set is similar to that mentioned in the above subsection. Both of these two images are labeled into five categories, including residential area, farmland, forest, grassland, and ocean, as shown in Figure 8. Each category contains approximately 2,000 SAR images and 2,000 optical images with 192 × 192 pixels per images.

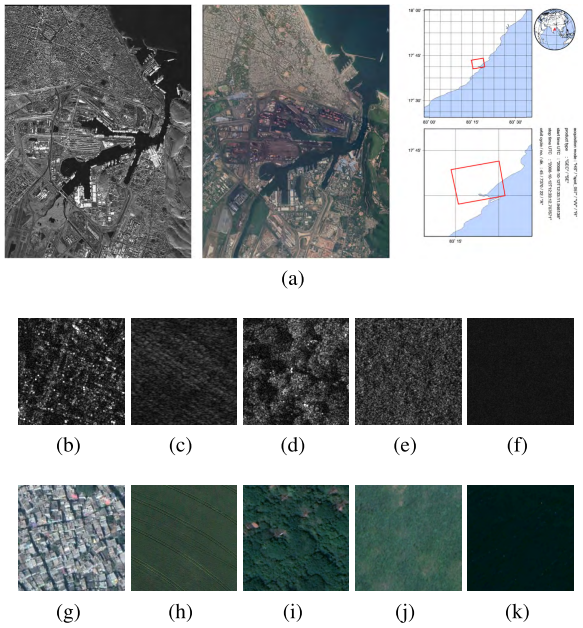

FIGURE 8. TerraSAR-X and optical images for the experiments. (a) From left to right are the original SAR image, the optical image of the same region and the geographical location of the experimental data. The second line shows the slices of the original SAR image, including (b) residential area. (c) farmland. (d) forest. (e) grassland. (f) ocean. (g)-(k) show the optical image slices of the same region.

D. OTHER DETAILS

For the source domain and target domain of each data set, we randomly selected 2/3 of them as the training set and the rest as the test set. To highlight the effectiveness of the algorithm in this paper, we applied no data augmentation tricks in the experiments.

To compare with the general neural networks, we directly utilized the whole CNN composed of the source encoder E_t and the classifier C which has been pre-trained on source domain with labels to classify the target domain images, which is considered to provide the baseline performance of the above three datasets. This baseline makes an equitable comparison by eliminating performance differences due to network architecture. In training, the time and space complexity of our method are about double of the baseline, because both E_s and E_t are required for computing. However, There's no more time and space complexity of our method than the baseline in the test, as their test architectures are the same.

We also compared our method with some other existing domain adaptation method, such as deep domain confusion(DDC) [24], deep adaptation networks(DAN) [26] and adversarial discriminative domain adaptation(ADDA) [25]. Moreover, we fine-tune the pre-trained CNN in the target domain with 30% and 100% of labeled training samples to compare our method with the supervised methods. These methods are also based on the same test network architecture.

IV. RESULTS AND DISCUSSIONS

A. PERFORMANCE EVALUATION

First of all, we will evaluate the performance of the proposed method and some of the comparison methods in the above three data sets. The performance of each method in each experiment is recorded in TABLE 4 with the form of $\mu \pm \sigma$, where μ is the mean test accuracy of the last ten epochs while σ is the standard deviation. To simplify, SimuX and SimuC represent simulated data while MeasX and MeasC represent measured data.

It is easy to find that the supervised CNN trained directly in the whole target domain is still the powerful method in the experiments. However, MLADA is very close to it and outperforms other methods in most of the experiments. Especially for the experiments SimuC→SimuX, SimuX→SimuC, and MeasX→MeasC, our method achieves the accuracy of more

than 99% and surpasses the supervised CNN that is trained using 30% labeled samples of the target training set. It should be noted that training CNN with a small number of labeled samples leads to poor performance in the experiments, SimuC→SimuX and SimuX→SimuC. We speculate that the performance degradation is caused by the imbalance of labeled training sample. In other words, the labeled samples do not cover all azimuths of the target, which lead to the overfitting of the supervised CNN.

Moreover, TABLE 5-9 represent the classification results for the five experiments of our method. For the two experiments in the simulation target data set, it can be observed that the misjudged categories are more likely to be classified as the trailer. While for the two experiments in the measured scene data set, the misjudged categories are more likely to be classified as cities. This phenomenon may be mainly caused by the data set bias between the training set and the test set. For the experiment of Optical→SAR, our method is successful in the categories of cities, forests, and oceans, but fails in farmlands and grasslands. It can be observed from the samples shown in Figure 8 that both classes of misjudgments lack texture information. The neural network can only learn color information from these two classes of scenes that are generally not suitable for the classification of grayscale SAR images.

TABLE 5. Confusion matrix for the recognition results. (SimuX→SimuC).

Category	bus	car	trailer	jeep	roller	truck
bus	1195	0	3	0	0	2
car	0	1200	0	0	0	0
trailer	0	0	1200	0	0	0
jeep	0	0	0	1200	0	0
roller	0	0	2	0	1198	0
truck	0	0	0	0	0	1200

TABLE 6. Confusion matrix for the recognition results. (SimuC→SimuX).

Category	bus	car	trailer	jeep	roller	truck
bus	1195	0	4	0	0	1
car	0	1200	0	0	0	0
trailer	0	0	1200	0	0	0
jeep	0	0	3	1197	0	0
roller	0	0	3	0	1197	0
truck	0	0	0	0	0	1200

TABLE 7. Confusion matrix for the recognition results. (MeasX→MeasC).

Category	cities	desert	farmland	mountain	ocean
cities	1150	0	0	0	0
desert	4	1146	0	0	0
farmland	13	0	1137	0	0
mountain	3	0	0	1147	0
ocean	0	0	0	0	1150

Considering that the electromagnetic wave is disturbed by noise during transmission in practice, we also add different levels of noise to the simulated echo. An additional experiment was conducted to check the anti-noise performance of our method. Meanwhile, we also compare MLADA with the supervised CNN directly trained in the source domain and target domain (30% and 100% of samples). The test accuracy of each object under different SNR conditions are recorded in TABLE 10-19. And the average accuracy of each method is shown in 9a and 9b for an intuitive comparison.

As we can see from the results, the performance of MLADA is similar to that of the supervised CNN trained using the whole target domain samples, when the SNR is greater than 15db. Then the accuracy decreases obviously

TABLE 8. Confusion matrix for the recognition results. (MeasC→MeasX).

Category	cities	desert	farmland	mountain	ocean
cities	1150	0	0	0	0
desert	8	1142	0	0	0
farmland	16	0	1128	0	6
mountain	13	0	0	1136	1
ocean	1	0	0	0	1149

TABLE 9. Confusion matrix for the recognition results. (Optical→SAR).

Category	cities	farmland	forest	grassland	ocean
cities	644	0	16	0	0
farmland	0	0	0	6	654
forest	8	0	652	0	0
grassland	0	0	203	0	457
ocean	0	0	0	0	660

TABLE 10. Test accuracy under noise condition. (SimuX→SimuC, 30db).

Categoy	CNN(source)	CNN(tgt 30%)	CNN(tgt 100%)	MLADA(our)
bus	0.1206	0.7672	1.0	1.0
car	0.0	0.6404	1.0	0.9988
trailer	1.0	0.7094	1.0	1.0
jeep	0.0	0.6527	1.0	1.0
roller	0.0008	0.6133	1.0	1.0
truck	0.0025	0.8719	1.0	0.9823
Average	0.1873	0.7091	1.0	0.9969

TABLE 11. Test accuracy under noise condition. (SimuX→SimuC, 25db).

Categoy	CNN(source)	CNN(tgt 30%)	CNN(tgt 100%)	MLADA(our)
bus	0.0	0.8966	1.0	1.0
car	0.0	0.5616	1.0	0.9991
trailer	1.0	0.6157	1.0	0.9996
jeep	0.0	0.6724	0.9967	0.9926
roller	0.0	0.6466	0.9985	0.9991
truck	0.0	0.7340	1.0	0.9827
Average	0.1667	0.6878	0.9992	0.9950

TABLE 12. Test accuracy under noise condition. (SimuX→SimuC, 20db).

Categoy	CNN(source)	CNN(tgt 30%)	CNN(tgt 100%)	MLADA(our)
bus	0.0	0.9137	0.9987	0.9971
car	0.0	0.5369	1.0	0.9909
trailer	1.0	0.5813	1.0	0.9995
jeep	0.0	0.5222	1.0	0.9951
roller	0.0	0.6096	0.9987	0.9991
truck	0.0	0.7032	1.0	0.9872
Average	0.1667	0.6469	0.9996	0.9948

TABLE 13. Test accuracy under noise condition. (SimuX→SimuC, 15db).

Categoy	CNN(source)	CNN(tgt 30%)	CNN(tgt 100%)	MLADA(our)
bus	0.0	0.8571	1.0	0.9909
car	0.0	0.6083	0.9938	0.9605
trailer	1.0	0.5726	0.9987	0.9815
jeep	0.0	0.4199	0.9926	0.986
roller	0.0	0.5862	0.9987	0.9954
truck	0.0	0.6083	0.9971	0.9651
Average	0.1667	0.6087	0.9971	0.9799

TABLE 14. Test accuracy under noise condition. (SimuX→SimuC, 10db).

Categoy	CNN(source)	CNN(tgt 30%)	CNN(tgt 100%)	MLADA(our)
bus	0.0	0.6822	0.9839	0.5816
car	0.0	0.4766	0.9778	0.8621
trailer	1.0	0.5382	0.9593	0.7783
jeep	0.0	0.4199	0.8780	0.8132
roller	0.0	0.5271	0.9002	0.9474
truck	0.0	0.6158	0.9261	0.61
Average	0.1667	0.5433	0.9376	0.7654

TABLE 15. Test accuracy under noise condition. (SimuC→SimuX, 30db).

Categoy	CNN(source)	CNN(tgt 30%)	CNN(tgt 100%)	MLADA(our)
bus	0.9987	0.7660	1.0	0.9987
car	0.0086	0.5763	1.0	1.0
trailer	0.7684	0.8325	1.0	1.0
jeep	0.0	0.6712	1.0	0.9987
roller	0.1699	0.4409	1.0	1.0
truck	0.7684	0.6983	0.9987	0.9975
Average	0.4523	0.6642	0.9997	0.9992

TABLE 16. Test accuracy under noise condition. (SimuC→SimuX, 25db).

Categoy	CNN(source)	CNN(tgt 30%)	CNN(tgt 100%)	MLADA(our)
bus	0.0	0.8571	1.0	1.0
car	0.0	0.5776	1.0	1.0
trailer	1.0	0.7611	0.9988	1.0
jeep	0.0	0.6318	0.9988	0.9987
roller	0.3362	0.4507	1.0	1.0
truck	0.0	0.6859	1.0	0.9975
Average	0.2227	0.6607	0.9995	0.9994

TABLE 17. Test accuracy under noise condition. (SimuC→SimuX, 20db).

Categoy	CNN(source)	CNN(tgt 30%)	CNN(tgt 100%)	MLADA(our)
bus	0.0	0.8091	1.0	0.9655
car	0.0	0.5222	1.0	1.0
trailer	1.0	0.6071	0.9987	0.9963
jeep	0.0	0.5764	0.9963	0.9926
roller	0.0	0.5160	1.0	1.0
truck	0.0	0.5813	0.9987	0.7869
Average	0.1667	0.6020	0.9989	0.9569

with reducing SNR, but still better than the supervised CNN trained using 30% of target samples. However, MLADA is an unsupervised method that doesn't require any labels of the target domain. Moreover, the CNN which is trained only in source domain suffered severe performance degradation when the SNR is less than 30db, and all of the objects are identified as the same type (trailer). This phenomenon is known as the data set bias in the machine learning area.

TABLE 18. Test accuracy under noise condition. (SimuC→SimuX, 15db).

Categoy	CNN(source)	CNN(tgt 30%)	CNN(tgt 100%)	MLADA(our)
bus	0.0	0.6663	1.0	0.6822
car	0.0	0.5333	0.9926	0.9421
trailer	1.0	0.4569	0.9938	0.9064
jeep	0.0	0.4138	0.9852	0.8965
roller	0.0	0.3387	0.9963	0.8756
truck	0.0	0.6453	0.9963	0.6018
Average	0.1667	0.5090	0.9940	0.8174

TABLE 19. Test accuracy under noise condition. (SimuC→SimuX, 10db).

Categoy	CNN(source)	CNN(tgt 30%)	CNN(tgt 100%)	MLADA(our)
bus	0.0	0.4951	0.9877	0.5866
car	0.0	0.5764	0.9249	0.7709
trailer	1.0	0.3818	0.9741	0.6995
jeep	0.0	0.0037	0.9667	0.7894
roller	0.0	0.3929	0.9027	0.7709
truck	0.0	0.6552	0.9507	0.4876
Average	0.1667	0.4175	0.9511	0.6845

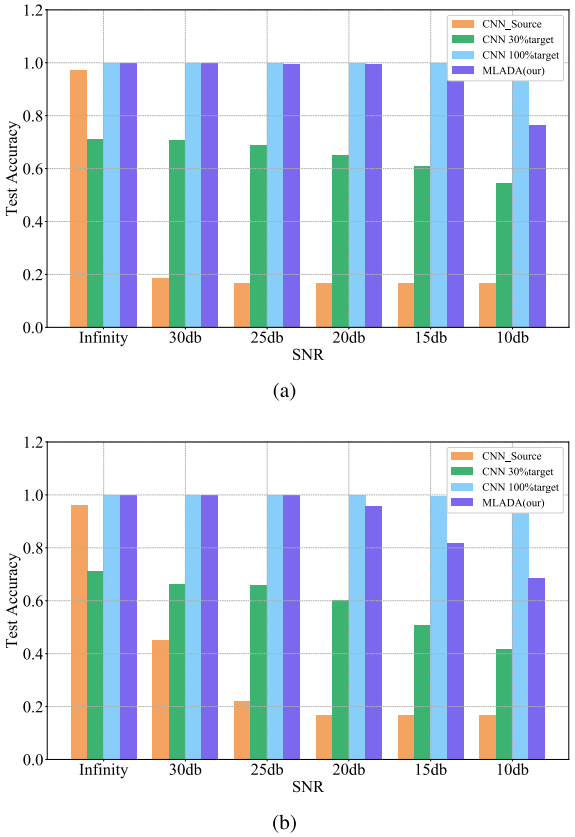

FIGURE 9. The recognition accuracy of each category varies with the SNR. (a) SimuX→SimuC. (b) SimuC→SimuX.

Figure 10 shows the feature maps of each subnetwork of a noisy input image. It can be noted that both MLADA and the supervised CNN trained in the whole domain data set produce clear feature maps with little noise. However, the feature maps of CNN trained in source domain is submerged in noise. This impact is becoming more pronounced in more

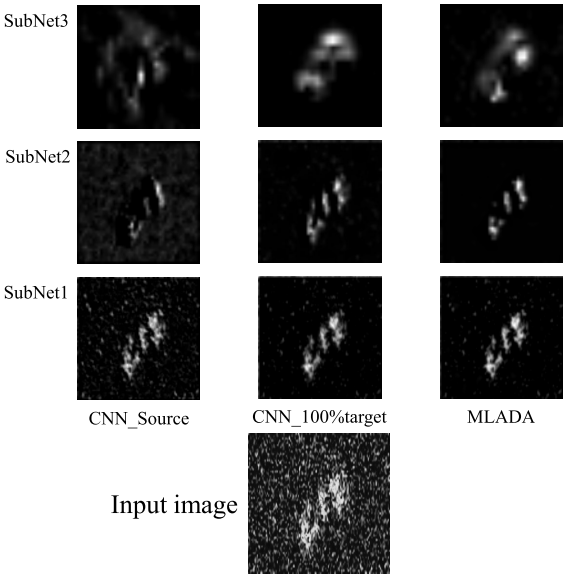

FIGURE 10. Output feature maps of each subnetwork.

discriminative high-level feature layers, which will result in a wrong classification.

B. DISCUSSION WITH VISUALIZATION

As described in section II, the source and target encoders extract features from the source and target domains, respectively. And then, all of these features are fed into the same classifier. Therefore, if our method is effective, the features extracted by E_t and E_s should obey the same distribution. To verify this conjecture, we use t-SNE [35] to project high-dimensional features into 2-dimensional cartesian coordinates, as shown in Figure 11. Although the 2-D projections cannot represent the higher-dimensional features absolutely, it can be roughly noted from the left column that the feature distributions of the target domains without MLADA are confused and quite different from those of the source domains. However, through domain adaptation using MLADA, the feature distributions of the target domains tend to be ordered and similar to the source domains, as shown in the right column of Figure 11. The results prove our conjecture.

Next, we will analyze the reliability of our algorithm with an explanatory visualization method. Recent researches show that in the error back-propagation of neural networks, the regions with a larger gradient on a feature map are usually assigned higher weights [36]- [37]. Thus, we can estimate the regions of interest of the neural network by calculating the derivative of a certain class score with respect to each pixel on the feature map. As the output of a convolutional layer usually contains many feature maps, the weighted sum of each feature map according to their gradients can roughly reflect the attention that the neural network paid to each region. And the specific implementation is given as follows:

$$p_i^k = \frac{1}{W \times H} \sum_{i=1}^{W} \sum_{j=1}^{H} \frac{\partial c_i'}{\partial F^k(w,h)}, \tag{20}$$

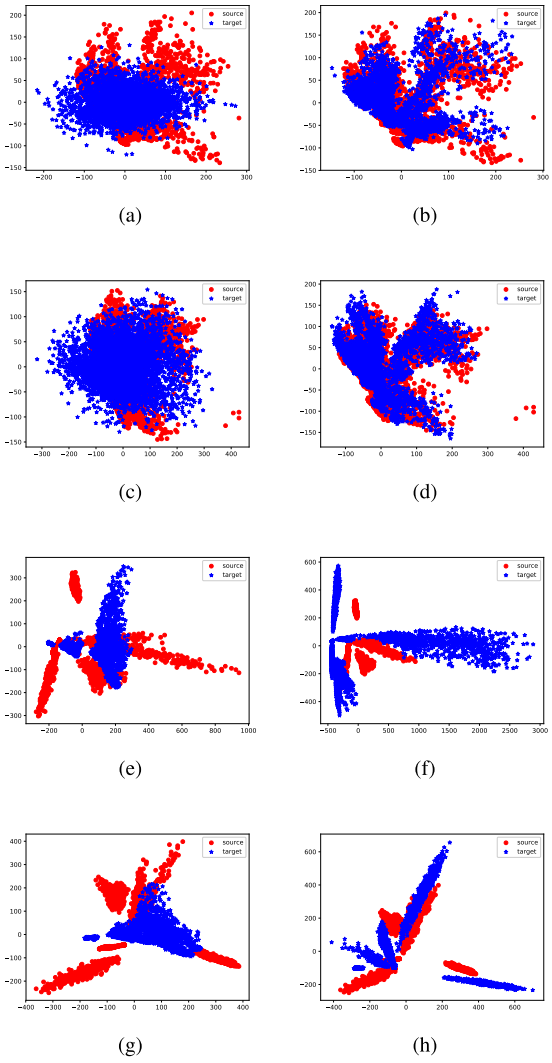

FIGURE 11. 2-D projection of features extracted by the source encoder E_S and target encoder E_t. (a)-(d) are projections of the simulated data, (e)-(h) are projections of the measured data. The left column represents projections before domain adaptation. The right column represents projections after domain adaptation.

$$L_i = ReLU(\sum_k p_i^k F^k), \tag{21}$$

where W and H represent the width and hight of a feature map, c_i' is the score of the ith class before normalized by the softmax function, $F^k(w,h)$ represents the activation value of position (w,h) in the kth feature map, and p_i^k is the weight of the kth feature map with respect to the ith category. The ReLU activation function is used to preserve only the positive part of the result. For more details, refer to literatures [28] and [37]. L_i produces a heat map of the same scale as the original image through bilinear interpolation to project the regions of interest to the original image. We check the results using the target domain images of all the experiments. The heat maps of regions of interest are illustrated in Figure 12.

As we can see from Figure 12, MLADA and the supervised CNN trained in the whole target domain highlight the similar

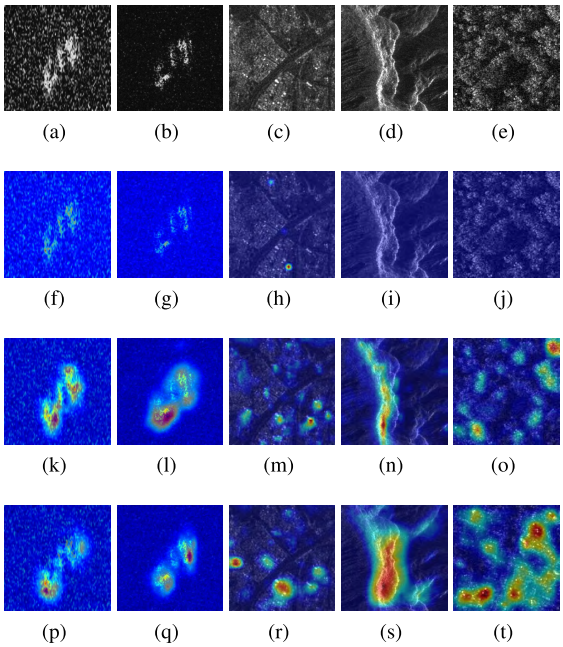

FIGURE 12. Regions of interest of the different for some target domain samples. (a)-(e) present samples of target domain images. (f)-(j) show the regions of interest of the CNN trained in source domain. (k)-(o) show the regions of interest of the CNN trained in target domain. (p)-(t) show the regions of interest of MLADA.

regions in the images, although the target CNN is more accurate due to the assistance of supervised information. However, the source CNN shows little or no interest in the target samples. The results provide strong support for the dependability of our method.

V. CONCLUSION

To overcome the problem that the neural network trained in a SAR data set of one frequency band cannot effectively classify images of another frequency band, we propose an adversarial domain adaptation method based on multi-level features named MLADA in this paper. It transfers the knowledge learned from one band images to the classification tasks of another band images with totally unsupervised settings. The experimental results of simulation data and measurement data show that our method's performance is superior to the general CNN trained in source domain or in target domain with insufficient labels. The experiments also present the strong anti-noise performance of MLADA. As long as the SNR is not too low, the test accuracies of MLADA are very close to the supervised CNN trained directly in the whole target domain. Moreover, we also visualize the regions of interest of our method, which proves the reliability of MLADA.

ACKNOWLEDGMENT

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

REFERENCES

[1] A. Hirose, Ed., *Complex-Valued Neural Networks: Advances and Applications*. Hoboken, NJ, USA: Wiley, 2013.
[2] 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.
[3] D. Guan, D. Xiang, G. Dong, T. Tang, X. Tang, and G. Kuang, "SAR image classification by exploiting adaptive contextual information and composite kernels," *IEEE Geosci. Remote Sens. Lett.*, vol. 15, no. 7, pp. 1035–1039, Jul. 2018.
[4] S. Ochilov and D. A. Clausi, "Operational SAR sea-ice image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 50, no. 11, pp. 4397–4408, Nov. 2012.
[5] M. Picco and G. Palacio, "Unsupervised classification of SAR images using Markov random fields and G_I^0 Model," *IEEE Geosci. Remote Sens. Lett.*, vol. 8, no. 2, pp. 350–353, Mar. 2011.
[6] Y. Sun, Z. 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.
[7] M. W. J. Davidson, E. Attema, N. Floury, B. Rommen, and P. Snoeij, "A closed-form expression relating classification accuracy to SAR system calibration uncertainty," *IEEE Geosci. Remote Sens. Lett.*, vol. 6, no. 3, pp. 467–470, Jul. 2009.
[8] B. Hou, B. Ren, G. Ju, H. Li, L. Jiao, and J. Zhao, "SAR image classification via hierarchical sparse representation and multisize patch features," *IEEE Geosci. Remote Sens. Lett.*, vol. 13, no. 1, pp. 33–37, Jan. 2016.
[9] C. He, T. Zhuo, D. Ou, M. Liu, and M. Liao, "Nonlinear compressed sensing-based LDA topic model for polarimetric SAR image classification," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 7, no. 3, pp. 972–982, Mar. 2014.
[10] X. Qin, H. Zou, S. Zhou, and K. Ji, "Region-based classification of SAR images using Kullback–Leibler distance between generalized gamma distributions," *IEEE Geosci. Remote Sens. Lett.*, vol. 12, no. 8, pp. 1655–1659, Aug. 2015.
[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, vol. 2012, pp. 1097–1105.
[12] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," Sep. 2014, *arXiv:1409.1556*. [Online]. Available: https://arxiv.org/abs/1409.1556
[13] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 770–778.
[14] S. Chen, H. Wang, F. Xu, and Y. Q. Jin, "Target classification using the deep convolutional networks for SAR images," *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 8, pp. 4806–4817, Aug. 2016.
[15] P. Zhong, Z. Gong, S. Li, and C.-B. Schönlieb, "Learning to diversify deep belief networks for hyperspectral image classification," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 55, no. 6, pp. 3516–3530, Jun. 2017.
[16] J. Wang, T. Zheng, P. Lei, and X. Bai, "Ground target classification in noisy SAR images using convolutional neural networks," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 11, no. 11, pp. 4180–4192, Nov. 2018.
[17] E. Keydel, S. Lee, and J. Moore, "MSTAR extended operating conditions: A tutorial," *Proc. SPIE*, vol. 2757, pp. 228–242, Jun. 1996.
[18] J. Su, D. V. Vargas, and K. Sakurai, "One pixel attack for fooling deep neural networks," *IEEE Trans. Evol. Comput.*, to be published. doi: 10.1109/TEVC.2019.2890858.
[19] Z. Huang, Z. Pan, and B. Lei, "Transfer learning with deep convolutional neural network for SAR target classification with limited labeled data," *Remote Sens.*, vol. 9, no. 9, p. 907, Aug. 2017.
[20] Y. Wang, C. Wang, and H. Zhang, "Combining single shot multibox detector with transfer learning for ship detection using Sentinel-1 images," in *Proc. SAR Big Data Era: Models, Methods Appl.*, Nov. 2017, pp. 1–4.
[21] Q. Song and F. Xu, "Zero-shot learning of SAR target feature space with deep generative neural networks," *IEEE Geosci. Remote Sens. Lett.*, vol. 14, no. 12, pp. 2245–2249, Dec. 2017.
[22] E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, and T. Darrell, "Deep domain confusion: Maximizing for domain invariance," Dec. 2014, *arXiv:1412.3474*. [Online]. Available: https://arxiv.org/abs/1412.3474
[23] M. Long, Y. Cao, J. Wang, and M. I. Jordan, "Learning transferable features with deep adaptation networks," Feb. 2015, *arXiv:1502.02791*. [Online]. Available: https://arxiv.org/abs/1502.02791

[24] M. Long, Y. Cao, J. Wang, and M. I. Jordan, "Deep transfer learning with joint adaptation networks," in *Proc. Int. Conf. Mach. Learn.*, vol. 70, 2017, pp. 2208–2217.
[25] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 2672–2680.
[26] H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, and M. Marchand, "Domain-adversarial neural networks," Dec. 2014, *arXiv:1412.4446*. [Online]. Available: https://arxiv.org/abs/1412.4446
[27] K. Bousmalis, G. Trigeorgis, N. Silberman, D. Krishnan, and D. Erhan, "Domain separation networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 343–351.
[28] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual explanations from deep networks via gradient-based localization," in *Proc. IEEE Int. Conf. Comput. Vis.*, Oct. 2017, pp. 618–626.
[29] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson, "Understanding neural networks through deep visualization," Jun. 2015, *arXiv:1506.06579*. [Online]. Available: https://arxiv.org/abs/1506.06579
[30] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York, NY, USA: Springer-Verlag, 2006.
[31] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," Feb. 2105, *arXiv:1502.03167*. [Online]. Available: https://arxiv.org/abs/1502.03167
[32] 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.
[33] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," Dec. 2014, *arXiv:1412.6980*. [Online]. Available: https://arxiv.org/abs/1412.6980
[34] 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.*, Dec. 2015, pp. 1026–1034.
[35] L. van der Maaten, "Barnes-hut-sne," Jan. 2013, *arXiv:1301.3342*. [Online]. Available: https://arxiv.org/abs/1301.3342
[36] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net," Dec. 2014, *arXiv:1412.6806*. [Online]. Available: https://arxiv.org/abs/1412.6806
[37] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 2921–2929.

WEI ZHANG was born Qingdao, China, in 1995. He received the B.E. degree in electronics from Beihang University, Beijing, China, in 2017. He is currently pursuing the M.S. degree in information and communication engineering with the National University of Defense Technology (NUDT), Changsha, China. His research interests include the application of transfer learning and deep learning algorithms to automatic radar target recognition.

YONGFENG ZHU received the B.E. degree in electronic engineering from Zhejiang University (ZJU), Hangzhou, China, in 2001, and the Ph.D. degree in information and communication engineering from NUDT, in 2009, where he is currently an Associate Research Fellow with the ATR Laboratory. His research interests include automatic target recognition technology, radar signal, and data processing.

QIANG FU was born in Changsha, China, in 1962. He received the B.E. degree in electronics engineering and the Ph.D. degree in information and telecommunication systems from the National University of Defense Technology (NUDT), Changsha, in 1983 and 2004, respectively.

He is currently a Professor with the ATR Laboratory, NUDT. His research interests include radar system design, precise guidance, and target recognition.