Progressive Data Augmentation Method for Remote Sensing Ship Image Classification Based on Imaging Simulation System and Neural Style Transfer

Qi Xiao, Bo Liu, Zengyi Li, Wei Ni, Zhen Yang, and Ligang Li

Abstract—**Deep learning has shown great power in processing remote sensing data, especially for fine-grained remote sensing ship image classification. However, the lack of a large amount of effective training data greatly limits the performance of neural networks. Based on current data augmentation methods, images of ships on the sea generated for remote sensing have the problem of distortion, blurring, and poor diversity. To tackle this problem, we propose a novel progressive remote sensing ship image data augmentation method that combines ship simulation samples and a neural style transfer (NST) based network to generate a large amount of transferred remote sensing ship images. Our method consists of two stages. The first stage uses a visible light imaging simulation system to generate ship simulation samples through three-dimensional models of real images. This stage can significantly increase the diversity of the training dataset. For the second stage, to eliminate the domain gap between real ship images and ship simulation samples, a few real images and a newly designed NST-based network called *Sim2RealNet* are employed to realize style transfer from simulation samples to real images. The proposed method was applied to a variety of ship targets to verify its effectiveness compared to other data augmentation methods on remote sensing image classification tasks. The experimental results demonstrate the effectiveness of the proposed method.**

Index Terms—**Domain gap, image classification, neural style transfer (NST), remote sensing, ship simulation samples.**

I. Introduction

With the rapid improvement of computer technology and GPU computing power, given sufficient data, deep learning has shown a strong dominance in the field of computer vision, such as image classification [1]–[3], object detection [4]–[6], and semantic segmentation [7], [8]. Following the success of deep learning and the increasing availability of remote sensing data, deep learning has been playing an increasingly important role in the field of remote sensing. With sufficient remote sensing data for training, researchers have focused on designing convolutional neural networks (CNNs) to perform feature selection [9]–[12], extraction [13], [14], and coding [15] on high-resolution remote sensing images, thereby improving network performance. Meanwhile, remote sensing data also bring unprecedented challenges to deep learning. For example, many application domains do not have access to big data, such as images of targeted ships, which are difficult to capture. At the same time, the labeling of remote sensing images requires considerable human and material resources. A lack of sufficient data will lead to the CNN overfitting problem.

On the one hand, from the model's architecture itself, many standard techniques have been proposed to alleviate the overfitting problem in the case of insufficient data. Nitish *et al.* [17] proposed the dropout method, which is a regularization technique that allows the network to learn more robust features by randomly zeroing some of the neuron values during the CNN training process. Another regularization technique is batch normalization [18], which normalizes the activation value of each layer into a vector with mean 0 and variance 1 to improve the stability and training speed of the neural network. This technique has now become a standard data preprocessing method for neural networks. Transfer learning [19], [20], pretraining [21], and one-shot [22] and zero-shot [23] learning algorithms can also effectively relieve the overfitting problem of deep neural networks.

On the other hand, in terms of training data to solve the overfitting problem, researchers have proposed the concept of data augmentation, a data-space solution to the problem of limited data that aims to intelligently augment the quality and quantity of the original small amount of data through a suite of techniques to improve deep neural network performance. Data augmentation can generate the most possible effective data from any amount of available data. Currently, the data augmentation methods most commonly used and considered most effective include traditional transformation, deep-learning-based, and imaging simulation system based methods. The first type mainly generates new samples by, e.g., rotating, cropping, and adding random noise to the original images [24].

The second type mainly encodes and decodes the input images to generate new images using neural networks, such as generative adversarial networks (GANs), which were first introduced by [25]. The third type uses imaging simulation systems to

Manuscript received June 29, 2021; revised August 10, 2021 and August 18, 2021; accepted August 30, 2021. Date of publication September 2, 2021; date of current version September 22, 2021. *(Corresponding author: Ligang Li.)*

Qi Xiao and Bo Liu are with the Key Laboratory of Electronics and Information Technology for Space System, National Space Science Center, Chinese Academy of Sciences, Beijing 100190, China, and also with the School of Computer Science and Technology, University of Chinese Academy of Sciences, Beijing 100049, China (e-mail: xiaoqi19@mails.ucas.ac.cn; liubo183@mails.ucas.ac.cn).

Zengyi Li, Wei Ni, Zhen Yang, and Ligang Li are with the Key Laboratory of Electronics and Information Technology for Space System, National Space Science Center, Chinese Academy of Sciences, Beijing 100190, China (e-mail: lzy_1710121227@vip.henu.edu.cn; niwei@nssc.ac.cn; yangzhen@nssc.ac.cn; liligang@nssc.ac.cn).

Digital Object Identifier 10.1109/JSTARS.2021.3109600

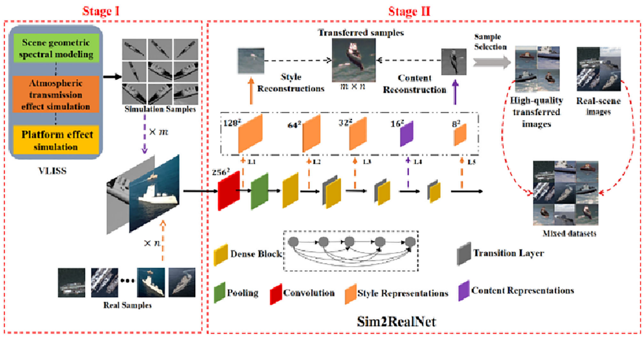


Fig. 1. Framework of our data augmentation method.

generate a large number of simulation samples and combine these samples with real images via CNNs [26]–[28].

In this article, we propose a novel progressive data augmentation method (Fig. 1) consisting of two stages. First, for a small number of real images of the targeted ship, we use a visible light imaging simulation system (VLISS) to generate m ship simulation samples using a 3-D model from an original dataset containing n ship images. Next, to eliminate the domain gap between the simulated samples and real images, which mainly manifests in the background of the two domains, we make each real image a style image and each ship simulation sample a content image, and we input them into *Sim2RealNet* to obtain $m \times n$ transferred samples. We select the k highest-quality transferred images using the structural similarity index measure (SSIM) algorithm and add them to the original dataset to constitute a mixed dataset. The mixed dataset is then used to train the CNN for image classification. We conducted extensive experimental verifications of the proposed method on remote sensing image classification tasks. The main contributions of this article can be summarized as follows.

1) We built a VLISS and utilized the system to generate a large number of simulation samples of a targeted ship for style transfer, and we designed *Sim2RealNet* to realize style transfer from the simulation samples to real images.
2) We introduced the SSIM [30] into our framework to achieve high-quality transferred image selection and *Sim2RealNet* model optimization.
3) We applied the proposed method to a variety of ship targets and conducted extensive experiments to demonstrate that our proposed method can achieve excellent performance and efficiency on fine-grained remote sensing ship image classification tasks.

II. Related Works

A. Data Augmentation Methods

Currently, there are three main categories of image augmentation methods. The first category consists of traditional white-box methods based on space and color manipulations, which can be divided into single-image and mixed-image methods. The second consists of black-box methods based on deep learning proposed in recent years, which include variational autoencoders (VAEs) [31] and GANs. The third has emerged over the past few years and consists of imaging simulation systems to generate simulation samples for data augmentation. This section describes each of these three categories in detail.

1) Traditional White-Box Data Augmentation Methods: Single-image augmentation methods perform all manipulations around a single image itself, which mainly includes geometric and color transformations, such as zooming in/out, flipping, cropping, rotation, noise injection, and grayscale. These methods are very easy to implement compared to other methods and were originally mainly applied on ImageNet [32] to increase the number of images for training the CNN.

In contrast, mixed-image augmentation refers to the simultaneous use of multiple images to generate one image. The synthetic minority oversampling technique [33] uses interpolation to generate new samples by sampling a small number of samples in the feature space of their neighboring samples to deal with the problem of sample imbalance, thereby improving classifier performance. However, mixed-image augmentation has problems such as marginalization and blindness when selecting neighbors.

Mixup [34] is a data augmentation method based on the principle of neighborhood risk minimization proposed by the Facebook Artificial Intelligence Research Institute and MIT in the article "Beyond Empirical Risk Minimization." It uses linear interpolation to obtain new sample data. Lusa *et al.* [35] proved that for high-dimensional data, the above two data augmentation algorithms have difficulty playing an effective role. Ioune *et al.* [36] proposed a method called sample pairing to realize data augmentation, which randomly selects two images and calculates the value of each pixel on each RGB channel of the two images as the pixel value of the new image. This method reduced the error rate from 8.22% to 6.93% on the CIFAR-10 dataset [37].

2) Deep Learning Black-Box Data Augmentation Methods: Among the data augmentation methods based on deep learning, the CNN has demonstrated a powerful feature extraction ability. A VAE maps an input image to a low-dimensional space through an encoder, learns the inherent encoding method of the image, and simultaneously uses a decoder to recover the image to the original size. Through the encoder and decoder, a VAE can generate new samples. The traditional VAE has a problem in that the generated images are relatively fuzzy. Although many studies have continued to improve its structure regarding this problem [38], because the VAE focuses on minimizing the interpolation between image pixels, there is still a lack of vivid image performance.

In 2014, Goodfellow [25] proposed GANs. By building GANs, the features of the original data can be flexibly supplemented, modified, and reconstructed to form a new feature vector with attributes similar to the original sample and generate a new sample. GANs are a two-player game between two networks called the discriminator and generator. The generator generates fake images from the given noise, and the discriminator effectively distinguishes whether the image is a real image or one generated by the generator. Owing to the constant competition between the discriminator and generator, the images generated by GANs constantly approach the real image. Finally, the discriminator can no longer distinguish whether an image comes from the real data or the generator. In deep convolutional GANs [39], both the discriminator and generator use a CNN to replace the multilayer perceptron in the traditional GANs. At the same time, to make the entire network differentiable, the pooling layer in the CNN is removed, and the fully connected layer is replaced with a global pooling layer to reduce the amount of calculation. Karras *et al.* [40] proposed progressive growing GANs to generate high-resolution images of size 1024×1024 for the first time. This method uses a progressive approach to generate images. First, a 4×4 image is generated. When the network is trained to a certain level, the resolution is gradually increased to 8×8, 16×16, and finally up to 1024×1024. In the field of remote sensing image generation, Lin *et al.* [41] designed multiple-layer feature-matching GANs to generate images containing simple objects, but the images were still blurred and distorted.

3) Imaging Simulation Data Augmentation Methods: With the development of imaging simulation systems and virtual engines, some attempts have been made in recent years to use simulation or virtual samples to train network models instead of real scene samples.

Li *et al*. [42] released a virtual image dataset. They used CityEngine and Unity3-D to construct a large-scale urban street scene to apply the virtual images to traffic vision research. Using transfer learning, Wang *et al*. [43] applied the simulation samples in an infrared band to the object detection task, which effectively improved the performance of the detector under the condition of extreme scarcity of real scene samples. In semantic segmentation research, Richter *et al*. [44] presented a method for building virtual datasets via modern video games and obtained the corresponding annotations using outside graphics hardware without access to the source code of the game. The above works applied virtual images directly as the training set to increase the feature diversity. However, ignoring the domain gap between simulation images and real scene images, when the number of simulation images is much larger than those of real images, the network will appear over-fitted to the simulated features. Nowadays, GANs are widely used to produce photorealistic synthetic images [45]; however, these images lack the corresponding annotations.

All of the above data augmentation methods have many limitations for a small number of remote sensing ship images. The traditional data augmentation methods can only increase the quantities of the dataset, and the increased data will easily cause the neural network to fall into overfitting. For deep learning data augmentation methods, the training of GANs relies on a large amount of data, and the generated images will have poor diversity and blurring problems. For imaging simulation data augmentation methods, the generated simulation samples and real images will often have large domain gaps, so the improvement of neural network performance is very limited. At the same time, the addition of simulation samples will reduce the accuracy of other types of target recognition. The proposed progressive data augmentation method can effectively solve these abovementioned problems as follows. In its first stage, the VLISS can generate a large number of ship simulation grayscale samples from different angles and under different environmental conditions through several 3-D models of real images. The structures and sizes of the simulated samples of these ships are strictly consistent with the real samples. At this stage, the simulation system can generate a large number of new simulation samples with diversity, which solves the problems of poor diversity, blurring, and distortion in current data augmentation methods. In the second stage, because there is a huge domain gap between the simulated samples and real images in their backgrounds, colors, and textures, the designed *Sim2RealNet* is employed to align the style-level features between the simulation samples and real images to realize style transfer from the simulation domain to the real domain, thereby improving the generalization ability of the network. We use style-aligned images to train the network and increase its robustness.

B. Neural Style Transfer

Due to the powerful feature extraction capabilities of the CNN, a style transfer network uses a CNN to render the content of an image into images of different styles. Style transfer networks have a wide range of applications, such as the generation of famous paintings and the migration of starry sky backgrounds. A style transfer network has two inputs: content image and style image. A style transfer network extracts the features of the style image at its shallow level as the style representation and extracts the features of the content image at its deep level as the content representation. Through the training of the neural network, the generated image is close to the style image in style and content image in content.

It is not appropriate to directly add the simulation samples generated by VLISS to the original dataset to train the neural network because the simulation samples and real images have two different domains. From the perspective of computer graphics, the information of an image can be divided into style and content information. The style information mainly includes the texture, background, and color of the image, and the content information includes the shapes, structures, and positions of the objects in the image. The shape, structure, and position of the targeted ship in a simulation sample and real image remain highly similar, so the content information of the two is essentially the same. However, for style information, the simulation samples generated by VLISS are single-channel grayscale images, which lack the rich style information of real remote sensing images. Therefore, the domain gap between the simulation samples and real images is mainly due to the large difference in style information. Inspired by the neural style transfer (NST) network, we use a CNN to extract the content features of the simulation samples and the style features of the real images and use a style transfer network to complete the style transfer from the simulated samples to the real images.

We made two improvements to this method. First, we used DenseNet-121 to replace VGG-19 [2] in the original method as the backbone. Second, we added SSIM loss to improve the loss function. We will introduce our designed *Sim2RealNet* in detail in Section III-B.

C. Structural Similarity Index Measure

The simplest and most widely used quality metrics are the mean squared error (MSE), computed by averaging the squared intensity differences of distorted and reference image pixels, and the related quantity, peak signal-to-noise ratio (PSNR). These metrics are appealing because they are simple to calculate, have clear physical meanings, and are mathematically convenient in the context of optimization. However, they are not very well matched with perceived visual quality. In the last three decades, a great deal of effort has gone into the development of quality assessment methods that take advantage of known characteristics of the human visual system. Most proposed perceptual quality assessment models have followed a strategy of modifying the MSE metric so that errors are penalized in accordance with their visibility. Image quality evaluation plays a very important role in remote sensing image processing. For remote sensing images captured by satellites, image quality evaluation can be used to measure the performance of the camera. Moreover, it can be used to evaluate the performance of the image processing module carried by the satellite.

SSIM is an evaluation metric used to measure the similarity between two images. It was first proposed by the Laboratory for Image and Video Engineering of the University of Texas at Austin in 2004 [30]. The SSIM algorithm evaluates the similarity of two images according to the following equation:

$$\mathrm{SSIM}_{(x,y)} = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{1}$$

where x and y are input images, μ_x and μ_y are the mean values of *x* and *y*, δ_x^2 and δ_y^2 are variances of *x* and *y*, respectively, and δ_{xy} is the covariance of *x* and *y*. SSIM $_{(x,\ y)}$ takes values in the range of -1 to 1. When images *x* and *y* are highly similar, SSIM $_{(x,\ y)} = 1$; otherwise, SSIM $_{(x,\ y)} = -1$.

SSIM measures the similarity between two images in terms of image brightness, contrast, and covariance. Therefore, using SSIM to measure the similarity between the image *Gen* generated by the style transfer network and the input simulation sample *Sim* is very effective. We use the SSIM value to achieve both high-quality generated sample selection and loss function improvement of *Sim2RealNet*.

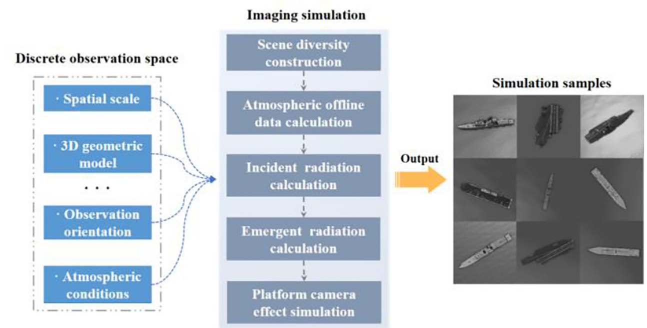


Fig. 2. Framework of VLISS. We organized imaging parameters (e.g., spatial size, 3-D geometric model, observation orientation, atmospheric conditions, etc.) into a discrete observation space (left). These imaging parameters are used as the input of the VLISS. The imaging simulation module (middle) contains the main steps to complete the full link simulation modeling and generates a large number of high-quality simulation images (right).

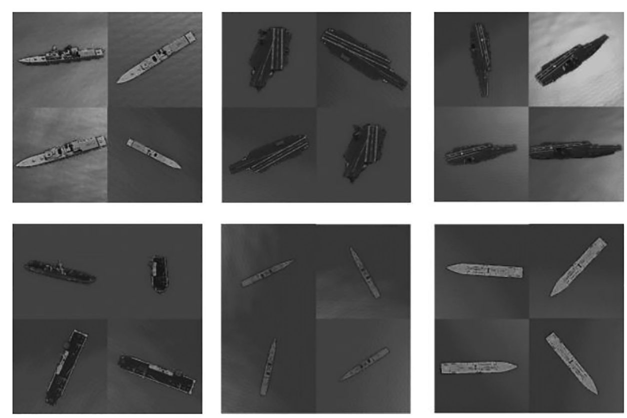


Fig. 3. Some simulation results for six target ships using our VLISS. From top left to bottom right: AS, KI, KU, LZ, MU, and SA.

III. Methods

A. VLISS

Based on the analysis of the visible light imaging radiation process and the mechanism of radiation transmission, we arrange the parameters that affect the characteristics of radiation transmission, including spatial scale, observation orientation, and atmospheric conditions, in order of importance.

These parameters are organized into a discrete observation space as the input of the imaging simulation framework. At the same time, combined with the simulation principle of visible light imaging, the simulation of samples is conducted via the following steps (Fig. 2). We performed imaging simulations of six types of ships. The number of real images of these ships was very low. Fig. 3 shows several of the simulation results of the visible light band under the conditions of 15° solar altitude, mid-latitude summer atmospheric model, and marine aerosol type. The squint distance is the distance between the camera and the ship target. The observation angle is determined by the observation altitude and azimuth angles. The observation altitude angle is the angle between the optical axis of the starting camera and the vertical axis at sea level. The observation azimuth angle is the angle between the projection at sea level and the direction of the bow (positive in the clockwise direction and negative in the anticlockwise direction). Table I lists the detailed simulation parameters of all the simulation samples shown in Fig. 3.

B. Sim2RealNet

This section describes the architecture of the proposed network in detail. The design of *Sim2RealNet* strongly follows the rules of NST and borrows insights from [46].

In the following, we compare each step of our network to those of another NST network. The full architecture of *Sim2RealNet* is illustrated in Fig. 4. To easily understand our work, we briefly summarize the NST algorithm by [46]. Given style image *S* and content image *C*, the NST network produces an output image *O*, which appears as *S* in style and *C* in content. The algorithm

TABLE I
DETAILED SIMULATION PARAMETERS CORRESPONDING TO FIG. 3

Ship	Sample no.	Simulation Parameters		
		Squint distance (km)	Observation azimuth (°)	Observation altitude angle (°)
Asagri	1	1.53	87.3	88.2
	2	1.37	62.5	90.0
	3	1.10	67.8	89.1
	4	2.05	-42.4	90
Kittyhawk	1	1.12	15.7	83.1
	2	1.66	-33.7	90
	3	1.53	58.9	90
	4	1.37	31.5	72.6
Kuznetsov	1	1.66	-3.4	90.0
	2	1.12	-127.8	88.2
	3	1.12	-99.1	90.0
	4	1.34	-90.0	90.0
Lzumo	1	1.56	-66.2	80.2
	2	1.42	0.00	82.2
	3	1.66	37.8	90.0
	4	1.68	137.8	90.0
Murasame	1	1.13	-138.4	90.0
	2	1.51	166.2	90.0
	3	1.24	-159.2	90.0
	4	1.33	72.1	90.0
Sacramento	1	1.22	92.1	90.0
	2	1.17	46.5	90.0
	3	1.19	90.0	90.0
	4	1.22	-45.2	90.0

There are four simulation samples for each type of ship: 1 (top left), 2 (top right), 3 (bottom left), and 4 (bottom right).

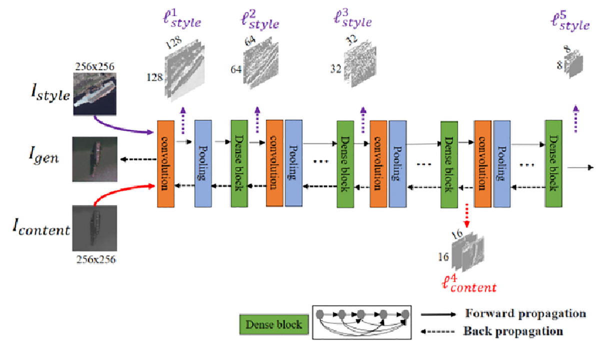
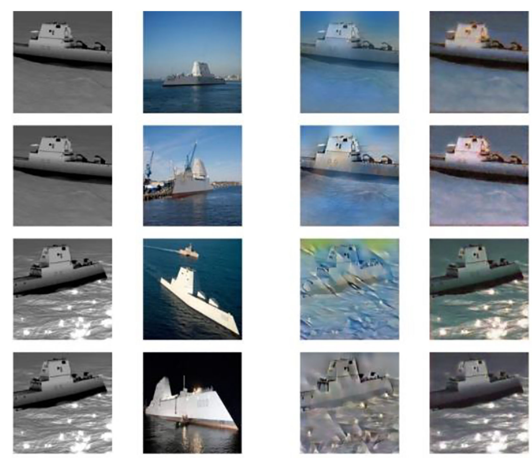


Fig. 4. Architecture of *Sim2RealNet*. I_{style} represent a real image, I_{content} represents a simulation sample, and I_{gen} represents a transferred image generated by *Sim2RealNet*. ℓ^i_{style} represents the *style loss* of the ith layer and ℓ^i_{content} represents the *content loss* of the ith layer.



(a) (b) (c) (d)

Fig. 5. Comparison of DenseNet against VGG-19 as our feature extractor. (a) Content image. (b) Style image. (c) VCG-19. (d) Ours: DenseNet-121.

minimizes the objective function

$$\mathcal{L}_{\text{total}} = \sum_{l=1}^{L} \alpha_l \mathcal{L}^l_{\text{content}} + K \sum_{l=1}^{L} \beta_l \mathcal{L}^l_{\text{style}} \tag{2a}$$

with

$$\mathcal{L}^l_{\text{content}} = \frac{1}{2N_l D_l} \sum_{ij} \left(F_l[O] - F_l[C]\right)^2_{ij} \tag{2b}$$

$$\mathcal{L}^l_{\text{style}} = \frac{1}{2N_l^2} \sum_{ij} \left(G_l[O] - G_l[S]\right)^2_{ij} \tag{2c}$$

where L represents the number of convolutional layers used to extract image features and l represents the lth convolutional layer. N_l is the number of filters in each convolutional layer and D_l represents the size of the feature map in each layer. The Gram matrix $G_l[.] = F_l[.]F_l[.]^T$ is defined as the inner product of the feature maps. α_l and β_l are the weights of each layer to configure the layer preferences. K is a weight that balances *content loss* (2b) and *style loss* (2c).

Network architecture: Among the previous NST networks, most of them [47]–[50] continue to employ the pretrained VGG-19 network [2] proposed in 2014 as the feature extractor, whereas we employ the pretrained DenseNet-121 [29] proposed in 2018 to replace VGG-19 in the original method as our backbone network. DenseNet is a CNN with dense connections. In this network, there is a direct connection between any two layers; that is, the input of each layer of the network is the union of the outputs of all previous layers, and the feature map learned by this layer will also be directly passed to all the following layers as input. In Fig. 5, the results show that employing DenseNet as our feature extractor can achieve better style transfer than using VGG-19.

DenseNet-121 consists of 121 layers, including four dense blocks. We refer the reader to [26] for more details on DenseNet-121; this article does not describe it in detail. As shown in Fig. 4, we selected the output features of five layers in DenseNet to build the feature extractor, which is the output after the first convolutional layer L1 and the output of the four dense blocks L2, L3, L4, and L5.

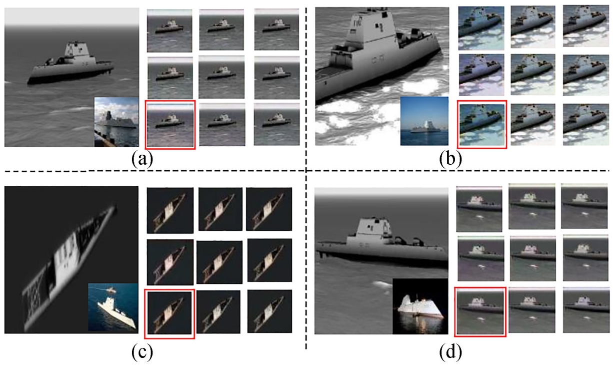


Fig. 6. Using different layers to build content representation and style representation. In each row, from left to right, we use *(L4), (L5),* and *(L4, L5)* to build the content representation. In each column, from top to bottom, we use *(L1, L2, L3), (L1, L2, L3, L4),* and *(L1, L2, L3, L4, L5)* to build style representation. The highest average $SSIM_{(O,G)}$ of the transferred images marked with a red box is equal to **0.909**.

Loss Function: In *Sim2RealNet*, for the network to realize style transfer from a simulation sample to a real image faster, the output image *O* is initialized to content image *C*, which is the simulation sample. At the same time, we also introduced a new *ssim loss* as follows:

$$\mathcal{L}_{\text{ssim}} = \text{SSIM}\,(O, C)\,. \tag{3}$$

Because *O* is initialized to *C*, at the beginning of the network, we have $\text{SSIM}(O,C)\approx 1$, and we must assign the network a larger penalty loss. With the training of the network, $\text{SSIM}(O,C)$ decreases alongside $\mathcal{L}_{\text{ssim}}$ when *O* gradually appears to be an image *I* in style, which is in line with our expectations.

We formulate the *Sim2RealNet* objective function by combining all three components as follows:

$$\mathcal{L}_{total} = \sum_{l=1}^{L} \alpha_l \mathcal{L}_c^l + K \sum_{l=1}^{L} \beta_l \mathcal{L}_s^l + W\mathcal{L}_{\text{ssim}} \tag{4}$$

where *W* is a weight that balances ssim *loss*.

Implementation details: We now describe the implementation details of *Sim2RealNet*. We employed pretrained DenseNet-121 as the feature extractor. As mentioned above, we selected the output of five layers (*L1–L5*) in DenseNet-121 for content and style feature representations. The effects of using different layers to build the content and style representations are illustrated in Fig. 6. We chose the highest $\text{SSIM}_{(O,G)}$ value, which was L4 as the content representation ($\alpha_l = 1$ for this layer and $\alpha_l = 0$ for other layers) and *L1, L2, L3, L4,* and *L5* as the style representations ($\beta_l = 1$ for these layers and $\beta_l = 0$ for other layers). Another important parameter is *W*; the effect of *W* is illustrated in Fig. 7. We chose the best result, $W = 25$, which has the highest $SSIM_{(O,G)}$ value. The other parameter $K = 1000$ for all the results. Our article is based on the implementation of [46]. Our code is available at.[1]

[1] [Online]. Available: https://github.com/xiaoqi25478/Progressive-Data-Augmentation-Method

Fig. 7. Effect of using different *W* values. We set ten groups of *W* parameters, from 0 to 45 in increments of 5 from left to right and top to bottom. The highest $\text{SSIM}_{(O,G)}$ of the transferred images marked with a red box is equal to **0.942**.

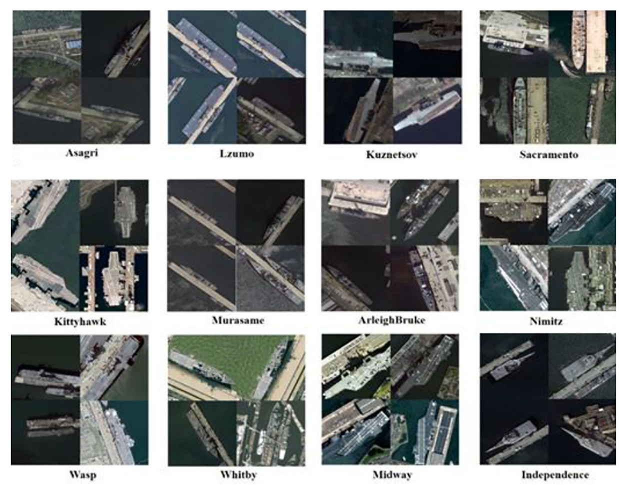


Fig. 8. Our original dataset consisting of 12 types of ships, of which six types (*AS*, *LZ*, *KU*, *SA*, *KI*, and *MU*) are scarce, and the remaining six types have an appropriate number.

IV. Experiments

In this section, we report our extensive experiments conducted to verify the effectiveness of our method compared with traditional and imaging simulation data augmentation methods on remote sensing ship image classification tasks. We used three datasets, each containing 12 types of ships: *Asagri* (*AS*), *Lzumo* (*LZ*), *Kuznetsov* (*KU*), *Sacramento* (*SA*), *Kittyhawk* (*KI*), *Murasame* (*MU*), *ArleighBruke* (*AR*), *Nimitz* (*NI*), *Wasp* (*WA*), *Whitby* (*WH*), *Midway* (*MI*), and *Independence* (*IN*).

A. Datasets

1) Real Image Dataset: To evaluate our method, we used FGSCR-42 [51], which is a public dataset for fine-grained ship classification in remote sensing images. The entire dataset contains approximately 9320 images, which are divided into 42 categories. The number of ships in each category varies significantly. To fully examine the performance of the proposed method, we selected six types of ships that are rare in number and applied our proposed data augmentation method to them. Considering the distribution of the numbers of each ship in the

TABLE II
SETTINGS OF THE ORIGINAL DATASET

Ship	Training set	Test set	Total
AS	35	35	70
LZ	32	31	63
KU	34	34	68
SA	25	25	50
KI	34	34	68
MU	32	31	63
AR	407	174	581
NI	388	165	553
WA	318	135	453
WH	195	83	278
MI	146	62	208
IN	148	62	210

TABLE III
SETTINGS OF THE SIMULATION SAMPLE DATASET

Ship	Training set	Test set	Total
AS	350	0	350
LZ	320	0	320
KZ	340	0	340
SA	250	0	250
KI	340	0	340
MU	320	0	320

TABLE IV
SETTINGS OF THE TRANSFERRED SAMPLE DATASET

Ship	Training set	Test set	Total
AS	350	0	350
LZ	320	0	320
KU	340	0	340
SA	250	0	250
KI	340	0	340
MU	320	0	320

Fig. 9. Some transferred results for six target ships obtained by *Sim2RealNet*. From top left to bottom right: AS, LZ, KU, SA, KI, and MU.

TABLE V
FULL LISTING OF ALL EXPERIMENTS

Method	ID	Operations	ID	Trad Augs	Sim Augs	Ours
Base	0	*no Augs*	30	√	-	-
Only Trad Augs	1	*random crop*	31	√	-	2× mixed
	2	*random horizontal flip*	32	√	-	4× mixed
	3	*cutout*	33	√	-	6× mixed
	4	*random erasing*	34	√	-	8× mixed
	5	*dual cutout*	35	√	-	10× mixed
	6	*mix up*	36	-	2× mixed	-
	7	*ricap*	37	-		2× mixed
	8	*cutmix*	38	-	4× mixed	-
	9	*label smoothing*	39	-		4× mixed
Only Sim Augs	10	1× mixed	40	-	6× mixed	-
	11	2× mixed	41	-		6× mixed
	12	3× mixed	42	-	8× mixed	-
	13	4× mixed	43	-		8× mixed
	14	5× mixed	44	-	10× mixed	-
	15	6× mixed	45	-		10× mixed
	16	7× mixed				
	17	8× mixed				
	18	9× mixed				
	19	10× mixed				
Only Ours	20	1× mixed				
	21	2× mixed				
	22	3× mixed				
	23	4× mixed				
	24	5× mixed				
	25	6× mixed				
	26	7× mixed				
	27	8× mixed				
	28	9× mixed				
	29	10× mixed				

real dataset, we also selected six types of ship targets with a larger number in the FGSCR-42 dataset. For these six types of ship targets, we did not apply any data augmentation methods. Overall, the 12 types of ships made up the original dataset for our experiments, as shown in Fig. 8.

As shown in Table II, we split the original dataset into a training set and test set with scientific proportions.

2) Simulation Sample Dataset: To compare our method with the data augmentation method based on imaging simulation, we

TABLE VI
RESULTS OF INDEPENDENT EXPERIMENT (ID 0 TO 29)

Method	ID	Operations	AP (%)												mAP (%)
			AS	**LZ**	**KU**	**SA**	**KI**	**MU**	AR	NI	WA	WH	MI	IN	
Base	0	*no Augs*	71.43	78.43	67.65	72.00	63.44	58.06	82.88	90.94	92.04	71.08	92.68	85.48	77.18
Only Trad Augs	1	*random crop*	75.86	74.19	67.65	76.00	79.41	64.52	85.63	96.36	100.00	85.54	100.00	88.71	82.82
	2	*random horizontal flip*	74.29	80.32	70.59	68.00	69.41	61.29	87.36	96.97	99.26	81.93	100.00	88.71	81.51
	3	*cutout*	74.29	74.19	73.53	64.00	64.71	59.61	89.66	94.55	100.00	74.70	96.77	93.55	79.96
	4	*random erasing*	80.00	82.33	73.53	72.00	76.47	64.19	86.78	94.55	99.26	81.93	100.00	88.71	83.31
	5	*dual cutout*	77.14	80.65	64.71	52.00	67.65	54.84	88.13	90.30	97.78	74.70	93.55	93.55	77.92
	6	*mix up*	62.86	70.97	66.98	44.00	68.24	58.06	86.78	86.06	94.81	66.27	87.10	85.48	73.13
	7	*ricap*	**73.13**	**80.65**	**76.47**	**72.00**	**79.41**	**67.74**	**89.53**	**91.52**	**100.00**	**81.93**	**100.00**	**91.94**	**83.69**
	8	*cutmix*	79.86	83.87	73.53	56.00	73.53	59.61	89.66	93.33	99.26	78.31	93.55	91.94	81.04
	9	*label smoothing*	80.00	83.87	61.76	56.00	70.59	51.61	88.51	91.52	100.00	77.11	96.77	91.94	79.14
Only Sim Augs	10	1× mixed	74.29	80.65	64.71	52.00	67.65	51.61	90.80	90.91	100.00	77.11	98.39	90.32	78.20
	11	2× mixed	77.14	83.55	70.59	52.00	79.41	67.74	91.95	92.12	100.00	75.90	100.00	91.94	81.86
	12	3× mixed	**80.00**	**80.32**	**76.47**	**76.00**	**72.35**	**64.52**	**94.25**	**92.73**	**100.00**	**80.72**	**100.00**	**88.71**	**83.84**
	13	4× mixed	82.86	81.29	78.24	60.00	68.35	75.97	91.38	92.73	100.00	78.31	98.39	91.94	83.29
	14	5× mixed	83.88	77.42	73.53	60.00	76.47	54.84	93.10	92.12	100.00	81.93	96.77	87.10	81.43
	15	6× mixed	82.86	80.65	64.71	60.00	67.35	61.29	93.10	93.33	100.00	83.13	100.00	88.71	81.26
	16	7× mixed	82.51	74.19	73.53	56.00	65.29	61.29	91.95	93.33	100.00	81.93	100.00	88.71	80.73
	17	8× mixed	81.33	83.87	72.35	72.00	70.59	67.74	87.36	96.97	100.00	79.52	100.00	90.32	83.50
	18	9× mixed	84.15	83.87	73.53	56.00	79.41	61.29	90.80	93.94	100.00	80.72	100.00	87.10	82.57
	19	10× mixed	80.00	77.42	64.71	56.00	76.47	54.84	92.53	92.73	98.52	81.93	96.77	85.48	79.78
Only Ours	20	1× mixed	88.03	87.10	83.53	56.00	70.59	67.54	90.23	89.70	100.00	78.31	96.77	88.71	83.04
	21	2× mixed	88.57	82.11	79.41	48.00	76.33	71.29	89.08	93.33	100.00	84.34	100.00	87.10	83.30
	22	3× mixed	85.00	85.63	73.53	52.00	77.01	76.55	89.66	88.48	99.26	79.52	98.39	90.32	82.95
	23	4× mixed	82.86	87.10	91.18	64.00	79.21	67.74	90.80	94.55	100.00	81.93	98.39	93.53	85.94
	24	5× mixed	87.14	84.19	70.59	60.00	73.53	75.54	87.93	94.55	99.26	81.93	98.39	91.94	83.75
	25	6× mixed	87.14	84.19	70.59	52.00	82.35	70.97	90.23	92.73	100.00	74.70	100.00	91.94	83.07
	26	7× mixed	**82.86**	**87.10**	**84.71**	**78.00**	**82.35**	**74.66**	**87.36**	**94.55**	**100.00**	**84.34**	**100.00**	**91.94**	**87.32**
	27	8× mixed	86.00	83.87	85.29	82.00	79.41	72.33	91.38	92.73	100.00	79.52	100.00	88.71	86.77
	28	9× mixed	87.14	90.32	82.53	52.00	79.41	72.12	89.66	93.33	99.26	78.31	98.39	90.32	84.40
	29	10× mixed	89.43	83.87	83.20	78.00	70.59	77.42	89.66	90.91	99.26	80.72	100.00	87.10	85.85

used VLISS to obtain simulation samples of six types of ships: *AS*, *LZ*, *KU*, *SA*, *KI*, and *MU*. The number of simulation samples for each ship was 10 times its number in the training set of the original dataset, as listed in Table III.

3) Transferred Sample Dataset: Similarly, for our progressive data augmentation method, we strictly took the following steps to generate transferred samples. First, for each of the six types of ships that require data augmentation, we used all the simulation samples in the simulation sample dataset and real images in the original dataset to perform one-to-one style transfer. Second, we used the $SSIM_{\ (sim,\ transferred)}$ value to measure the quality of each transferred sample. The larger the SSIM value, the better the transfer effect of the transferred sample is. Finally, we selected a certain number of transferred samples for each ship in the order of $SSIM_{\ (sim,\ transferred)}$ from high to low, and the number of transferred samples for each ship was ten times that in the training set of the original dataset, as listed in Table IV Fig. 9 shows some of the transferred samples for each ship obtained using *Sim2RealNet*.

B. Experimental Settings

To fully test the performance of our proposed method, we conducted 46 sets of experiments, divided into two types: independent and mutual experiments. In the independent experiments, we only used one data augmentation method for each experiment and compared it with other methods to illustrate the excellent performance of our method (IDs 0 to 29).

In the mutual experiments, each experiment used other data augmentation methods and our method at the same time to verify the compatibility of our method in cooperating with other data augmentation methods for CNN (IDs 30 to 45). As listed in Table V, for traditional data augmentation methods (*Trad Augs*), we used a variety of augmentation methods that are currently commonly used. These included *random crop*, *random horizontal flip*, *cutout*, *random erasing*, *dual cutout*, *mix up*, *ricap*, *cutmix*, and *label smoothing*. We set up a series of trapezoidal experiments using imaging simulation methods (*Sim Augs*) and our method (*Ours*). In the table, 1×mix means adding double the number of images, and so on, which are simulation samples

TABLE VII
RESULTS OF MUTUAL EXPERIMENT (ID 30 TO 45)

ID	Trad Augs	Sim Augs	Ours	AP (%)												mAP (%)
				AS	**LZ**	**KU**	**SA**	**KI**	**MU**	AR	NI	WA	WH	MI	IN	
30	√	-	-	78.65	87.87	74.02	72.00	77.65	64.88	88.33	93.94	94.81	79.21	94.11	87.21	82.72
31	√	-	2× mixed	85.71	87.10	77.18	88.00	87.12	67.74	89.08	98.79	94.81	89.16	100.00	93.55	**88.19**
32	√	-	4× mixed	88.57	90.06	79.41	84.00	100.00	74.19	81.61	99.39	94.07	83.13	98.39	90.32	**88.60**
33	√	-	6× mixed	79.57	91.33	79.41	76.00	97.06	74.19	84.48	96.97	97.78	87.95	98.39	98.39	**88.46**
34	√	-	8× mixed	80.00	89.10	77.43	80.00	94.12	70.97	83.33	95.15	97.78	84.34	98.39	95.16	**87.15**
35	√	-	10× mixed	83.18	89.87	85.48	74.00	94.12	64.52	88.51	96.97	92.59	84.34	100.00	83.87	**86.45**
36	-	2× mixed	-	77.14	83.55	70.59	52.00	79.41	67.74	91.95	92.12	100.00	75.90	100.00	91.94	81.86
37	-		2× mixed	**82.86**	**87.10**	**79.41**	**76.00**	**82.35**	**77.74**	**92.53**	**93.94**	**100.00**	**81.93**	**98.39**	**91.94**	**87.01**
38	-	4× mixed	-	82.86	81.29	78.24	60.00	68.35	75.97	91.38	92.73	100.00	78.31	98.39	91.94	83.29
39	-		4× mixed	**85.71**	**90.32**	**70.59**	**80.00**	**82.35**	**79.52**	**89.66**	**95.76**	**100.00**	**79.52**	**100.00**	**90.32**	**86.98**
40	-	6× mixed	-	82.86	80.65	64.71	60.00	67.35	61.29	93.10	93.33	100.00	83.13	100.00	88.71	81.26
41	-		6× mixed	**84.29**	**87.42**	**82.35**	**72.00**	**76.47**	**74.19**	**95.23**	**96.97**	**100.00**	**80.72**	**100.00**	**90.32**	**86.66**
42	-	8× mixed	-	81.33	83.87	72.35	72.00	70.59	67.74	87.36	96.97	100.00	79.52	100.00	90.32	83.50
43	-		8× mixed	**84.12**	**86.44**	**79.41**	**78.00**	**79.41**	**67.74**	**91.95**	**96.91**	**100.00**	**80.72**	**100.00**	**93.55**	**86.52**
44	-	10× mixed	-	80.00	77.42	64.71	56.00	76.47	54.84	92.53	92.73	98.52	81.93	96.77	85.48	79.78
45	-		10× mixed	**80.00**	**83.87**	**70.59**	**58.00**	**70.59**	**70.97**	**89.08**	**93.94**	**100.00**	**83.13**	**100.00**	**88.71**	**82.41**

(generated using the imaging simulation method) or transferred samples (generated using our method) to the training set in the original dataset. Note that only *AS*, *LZ*, *KU*, *SA*, *KI*, *and MU* have simulation samples and transferred samples, and the number of each type of ship in the test set was constant throughout.

C. Implementation Details

We conducted all experiments using PyTorch 1.7.0, CUDA 11.1, and Python 3.8.7, and we used an NVIDIA RTX 3090 GPU for training and testing. We used the average precision of each ship to measure the performances of all methods in this article.

For remote sensing ship image classification tasks, we used ResNet-34 [52] as our classification network, where we set the input image size to 256×256, learning rate to 0.003, and batch size to 16. Moreover, we adopted a mini-batch SGD optimizer [53] with a momentum of 0.9. We used the regularization method, and the weight decay was set to 1e-4. Each network was trained for 100 epochs.

For *Sim2RealNet*, we set the input image size to 256×256, learning rate to 0.01 and batch size to 64, and adopted the Adam optimizer [54] with a momentum of 0.3. We also used the regularization method, and the weight decay was set to 1e-3.

For traditional data augmentation, we used the following methods. The *random crop* method crops the training image randomly using a ratio of 0.6 to 1.0. *Random horizontal flip* is a type of image data augmentation that horizontally flips a given image with a given probability; we set its probability to 0.5. The *cutout* method randomly selects a fixed-size square area in the training image and then fills it with all zeros. Of course, to avoid the impact of zero-filled regions on training, the data should be centered and normalized. The size of the square we chose is 60 × 60. The *random erasing* method randomly selects a fixed-size square area in the training image and then replaces it with the average pixel value. We again chose a square size of 60 × 60. The principle of the *dual cutout* method is similar to that of *cutout*, but it simultaneously performs two random cutouts from the same image and then stitches them together. *The mixup method* mixes different images to expand the training dataset. The *ricap* method crops one part out of four images and recombines these parts into a new image. The *cutmix* method cuts off a part of the image but instead of filling the cut pixels with zeros, it randomly fills them with pixel values from other data in the training set such that the classification results are distributed according to a certain proportion. Finally, *label smoothing* narrows the gap between the minimum and maximum values in the label, which can counter the problem of neural network overfitting.

D. Experimental Results

We used mean average precision (mAP) as the evaluation method for all data augmentation methods in our experiments. The reason for this is that in the field of computer vision, mAP is a general evaluation method for computer vision tasks.

Table VI lists the results of the independent experiments. The experimental groups with ID 7, 12, and 16 represent the best results of the traditional methods, imaging simulation methods, and our method, respectively. Traditional data augmentation methods improve the mAP value of the remote sensing ship classification tasks by almost 6.5%. However, the traditional data augmentation methods can simply increase the quantity of data in the dataset, and the increased data can often cause the neural network to fall into overfitting. The imaging simulation data augmentation methods improve the mAP value by almost 6.6%. As shown in Table VI, as more simulation samples are added to the training set, the accuracy of the model begins to decrease. This indicates that the huge domain difference between the simulated sample and the real image will degrade the model's performance. Our proposed data augmentation method improves the mAP value by almost 10.1%, addressing the problems of the above two methods. The first stage, the imaging simulation system significantly increases the diversity of the training dataset. The second stage, which is the style transfer network to eliminate the domain gap between real ship images and ship simulation samples, *Sim2RealNet*, is employed to transfer the style from simulation samples to real images. The experimental results show that our method exhibits excellent performance as an independent data augmentation method.

Table VII lists the results of the mutual experiments. From the two adjacent rows of experimental results, it can be concluded that when used with other methods, our method can improve the accuracy of the model by 3%–7%. Our method substantially improves the performance of the CNN when compared with the improvements obtained by other data augmentation methods. However, it can also be seen that as the number of transferred samples gradually increases, it also causes a decrease in network performance. This shows that the transferred sample does not completely eliminate the domain gap between the simulated and real samples.

V. Conclusion

In this article, we proposed a progressive data augmentation method that combines simulation samples and NST for remote sensing ship image classification. Compared to other data augmentation methods, our method can effectively solve the problems of distortion, blurring, and poor diversity of the generated images. Extensive experimental results prove that, despite the scarcity of real images, our method can still effectively improve the accuracy of remote sensing ship classification tasks without reducing the recognition accuracy for other types of ships. We further expanded the application prospects of simulated images in the field of deep learning.

However, our method still has many disadvantages: the generated image resolution is too low, the time cost of style transfer is large, and the amount of augmented data generated still inevitably leads to the CNN overfitting problem. In future article, we will focus on solving these problems and aim to apply this method to remote sensing image target detection tasks.

References

[1] 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.

[2] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. IEEE Int. Conf. Learn. Representation*, 2015, Art. no. 14124313.

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.

[4] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2014, pp. 580–587.

[5] X. Sun, P. Wang, C. Wang, Y. Liu, and K. Fu, "PBNet: Part-based convolutional neural network for complex composite object detection in remote sensing imagery," *ISPRS J. Photogramm. Remote Sens.*, vol. 173, pp. 50–65, 2021.

[6] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 779–788.

[7] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 3431–3440.

[8] H. Noh, S. Hong, and B. Han, "Learning deconvolutional network for semantic segmentation," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 1520–1528.

[9] H. Gu, H. Li, L. Yan, Z. Liu, T. Blaschke, and U. Soergel, "An object-based semantic classification method for high resolution remote sensing imagery using ontology," *Remote Sens.*, vol. 9, no. 4, 2017, Art. no. 329.

[10] X. Sun, B. Wang, Z. Wang, H. Li, H. Li, and K. Fu, "Research progress on few-shot learning for remote sensing image interpretation," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 14, pp. 2387–2402, Jan. 2021.

[11] G. Liu; L. Gao, and L. Qi, "Hyperspectral image classification via multi-feature-based correlation adaptive representation," *Remote Sens.*, vol. 13, no. 7, 2021, Art. no. 1253.

[12] S. Xiang, Q. Xie, and M. Wang, "Semantic segmentation for remote sensing images based on adaptive feature selection network," *IEEE Geosci. Remote Sens. Lett.*, to be published, doi: 10.1109/LGRS.2021.3049125.

[13] W. Hu, Y. Huang, L. Wei, F. Zhang, and H. Li, "Deep convolutional neural networks for hyperspectral image classification," *J. Sensors*, vol. 2015, no. 258619, 2015, Art. no. 7954154.

[14] W. Liu *et al.*, "Accurate building extraction from fused DSM and UAV images using a chain fully convolutional neural network," *Remote Sens.*, vol. 11, no. 24, 2019, Art. no. 2912.

[15] Q. He, X. Sun, Z. Yan, and K. Fu, "DABNet: Deformable contextual and boundary-weighted network for cloud detection in remote sensing images," *IEEE Trans. Geosci. Remote Sens.*, to be published, doi: 10.1109/TGRS.2020.3045474.

[16] Q. Chen, L. Wang, Y. Wu, G. Wu, Z. Guo, and S. L. Waslander, "Aerial imagery for roof segmentation: A large-scale dataset towards automatic mapping of buildings," *ISPRS J. Photogram. Remote Sens.*, vol. 147, pp. 42–55, Jan. 2019.

[17] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

[18] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 448–456.

[19] K. Weiss, T. M. Khoshgoftaar, and D. Wang, "A survey of transfer learning," *J. Big Data*, vol. 3, 2016, Art. no. 9.

[20] L. Shao, F. Zhu, and X. Li, "Transfer learning for visual categorization: A survey," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 5, pp. 1019–1034, May 2015.

[21] D. Erhan, Y. Bengio, A. Courville, P. A. Manzagol, and P. Vincent, "Why does unsupervised pre-training help deep learning?," *J. Mach. Learn. Res.*, vol. 11, pp. 625–660, 2010.

[22] J. Pennington, R. Socher, and C. D. Manning, "GloVe: Global vectors for word representation," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2014, pp. 1532–1543.

[23] A. Halevy, P. Norvig, and F. Pereira, "The unreasonable effectiveness of data," *IEEE Intell. Syst.*, vol. 24, no. 2, pp. 8–12, Mar./Apr. 2009.

[24] J. Wang and L. Perez, "The effectiveness of data augmentation in image classification using deep learning," 2017, *arXiv:1712.04621*.

[25] I. J. Goodfellow *et al.*, "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, Art. no. 27.

[26] A. G. Howard, M. Zhu, B. Chen, and J. Sun, "Mobile-nets: Efficient convolutional neural networks for mobile vision applications," 2017, *arxiv.org/abs/1704.04861*.

[27] M. Sandler, A. Howard, M. L. Zhu, A. Zhmoginov, and L. C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 4510–4520.

[28] X. Y. Zhang, X. Y. Zhou, M. X. Lin, and J. Sun, "Shuffle-net: An extremely efficient convolutional neural network for mobile devices," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 6848–6856.

[29] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 4700–4708.

[30] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.

[31] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," 2013, *arXiv:1312.6114*.

[32] O. Russakovsky *et al.*, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, pp. 211–252, 2015.

[33] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *J. Artif. Intell. Res.*, vol. 16, no. 1, pp. 321–357, 2002.

[34] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "Mixup: Beyond empirical risk minimization," 2017, *arXiv:1710.09412*.

[35] L. Lusa, "SMOTE for high-dimensional class-imbalanced data," *BMC Bioinf.*, vol. 14, no. 1, 2013, Art. no. 106.

[36] I. Hiroshi, "Data augmentation by pairing samples for images classification," 2018, *arXiv:1801.02929*.

[37] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Tech. Rep., 2009, p. 5.

[38] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey, "Adversarial autoencoders," 2016, *arXiv:1511.05644*.

[39] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," 2015, *arXiv:1511.06434*.

[40] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of gans for improved quality, stability, and variation," 2017, *arXiv:1710.10196*.

[41] D. Lin, K. Fu, Y. Wang, G. Xu, and X. Sun, "MARTA GANs: Unsupervised representation learning for remote sensing image classification," *IEEE Geosci. Remote Sens. Lett.*, vol. 14, no. 11, pp. 2092–2096, Nov. 2017.

[42] X. Li, K. Wang, Y. Tian, L. Yan, and F.-Y. Wang, "The paralleleye dataset: Constructing large-scale artificial scenes for traffic vision research," in *Proc. IEEE 20th Int. Conf. Intell. Transp. Syst.*, 2017, Art. no. 24893706.

[43] Z. B. Wang, "Feasibility study on application of simulated images in deep learning," M.S. thesis, School Phys. Optoelectron. Eng., Xidian Univ., Xi'an, China, 2019.

[44] S. R. Richter, V. Vineet, S. Roth, and V. Koltun, "Playing for data: Ground truth from computer games," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 102–118.

[45] K. Wang, C. Gou, Y. Duan, Y. Lin, X. Zheng, and F.-Y. Wang, "Generative adversarial networks: Introduction and outlook," *IEEE/CAA J. Automatica Sinica*, vol. 4, no. 4, pp. 588–598, 2017.

[46] L. A. Gatys, A. S. Ecker, and M. Bethge, "Image style transfer using convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 2414–2423.

[47] F. Luan, S. Paris, E. Shechtman, and K. Bala, "Deep photo style transfer," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2017, pp. 4990–4998.

[48] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," 2016, *arXiv:1603.08155*.

[49] H. Zhang and K. Dana, "Multi-style generative network for Real-time transfer" in *Proc. Eur. Conf. Comput. Vis. Workshops*, 2018, pp. 1029–1037.

[50] Y. Li, M.-Y. Liu, X. Li, M.-H. Yang, and J. Kautz, "A closed-form solution to photorealistic image stylization," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 453–468.

[51] Y. Di, Z. Jiang, and H. Zhang, "A public dataset for fine-grained ship classification in optical remote sensing images," *Remote Sens.*, vol. 13, no. 4, 2021, Art. no. 747.

[52] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.

[53] P. Goyal *et al.*, "Accurate, large minibatch sgd: Training imagenet in 1 hour," 2017, *arXiv:1706.02677*.

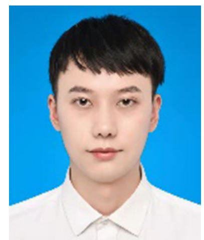
[54] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. IEEE Int. Conf. Learn. Representation*, 2015, Art. no. 6628106.

Qi Xiao received the B.E. degree in communication engineering from Xiangtan University, Xiangtan, China, in 2019. He is currently working toward the M.S. degree with the University of the Chinese Academy of Sciences, Beijing, China.

His research interests include domain adaptation and neural style transfer.

Bo Liu received the B.E. degree from the North University of China, Taiyuan, China, in 2018. He is currently working toward the Ph.D. degree in computer applications with the University of the Chinese Academy of Sciences, Beijing, China.

His research interests include domain adaptation and object detection based on simulation-scene images.

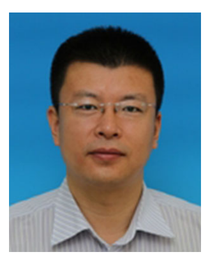


Zengyi Li received the B.E. degree in automation from Henan University, Kaifeng, China, in 2021. He is currently working toward the M.S. degree with the University of the Chinese Academy of Sciences, Beijing, China.

His research interests include deep learning and simulation system.

Wei Ni received the B.E. degree in automation from Northwest Normal University, Lanzhou, China, in 2011.

He is currently a Senior Engineer with the National Space Science Center, Chinese Academy of Sciences, Beijing, China.



Zhen Yang received the B.S. and M.S. degrees in communication and electronic engineering from National Defense University, Changsha, China, in 1994 and 1997, respectively, and the Ph.D. degree in applied computing from the University of the Chinese Academy of Sciences, Beijing, China, in 2014.

He is currently a Professor with National Space Science Center, Chinese Academy of Sciences. His research interests include complex system simulation, spatial task collaborative design and demonstration, spatial information services, and distributed space systems.

Ligang Li received the Ph.D. degree in optical engineering from the Chinese Academy of Sciences, Beijing, China, in 2006.

He is currently a Professor with the Chinese Academy of Sciences. His research interests include optical imaging simulation system technology, spatial information processing, and other research topics.