Cloud-Aware Generative Network: Removing Cloud From Optical Remote Sensing Images

Linjian Sun, *Student Member, IEEE*, Ye Zhang, *Member, IEEE*, Xuling Chang, Yanjie Wang, and Jiajia Xu

***Abstract*— In the optical remote sensing and earth observation fields, clouds severely obscure the land's visibility and degrade the image. In recent years, there have been many excellent efforts to mitigate the effects of cloud cover. However, it has been found that there will be some blurs in the area if a single degraded image is restored by autoencoder-based methods. This letter focuses on removing clouds from single optical remote sensing images by autoencoder-based methods without multitemporal information while at the same time mitigating blurs caused by missing information. Therefore, we propose a novel cloud removal method that combines image inpainting and image denoising, called the Cloud-Aware Generative Network (CAGN). The CAGN consists of two stages: the first stage is a recurrent convolution network for potential cloud region detection and the second is an autoencoder for cloud removal. The method uses a side-guided method that adds attention mechanisms in the first stage to assist in predicting the mask. Furthermore, to update the mask adaptively for restoring degraded image areas greedily, the method embeds partial convolution in the autoencoder to condition the convolution calculation of pixels in the regions of thick clouds at different layers. Extensive experiments demonstrate clearly that CAGN can easily achieve a considerable increase in the peak signal-to-noise ratio (PSNR) and the structural similarity index (SSIM) compared with a competitive baseline model.**

***Index Terms*— Attention mechanism, cloud detection, cloud removal, generative network, remote sensing.**

I. Introduction

REMOTE sensing technology has been widely used for earth observation. Optical remote sensing image processing is an important category, and it is easily affected by cloud cover. Clouds will severely hamper the visibility of the land and degrade an image considerably. Cloud cover can cause partial information loss, which will cause adverse effects on subsequent image processing. Therefore, solving the degradation problem of optical remote sensing image occluded by cloud cover has become an urgent need.

In the last few years, to mitigate the influence of cloud cover, many methods have been proposed. These methods can be classified into three categories [1]: inpainting-based methods, multispectral-based methods, and multitemporal-based methods. Inpainting-based methods [2], [3] make use of the context surrounding the region occluded by clouds to infer and complete the lost information in images. Owing to the context without clouds nearby, the cloud region is usually similar to the occluded part. Image inpainting is an important task in computer vision that aims to fill missing pixels of an image. The challenge mainly comes from synthesizing visually realistic and semantically plausible pixels for the missing region that are coherent with surrounding image content. Multispectral-based methods [4]–[6] use multispectral data for detecting the region of cloud and reconstructing the missing part of the image. Multitemporal-based methods [7], [8] mainly use the correlation of different temporal images to solve the problem of cloud removal; therefore, both temporal coherence and spatial coherence can be used. In addition, if considering different cloud cover conditions, the multispectral methods are mainly applied for thin-cloud removal, while multitemporal-based methods are more suitable for large-area thick-cloud removal. Recently, convolutional neural networks (CNNs) have been very successful and are used on a variety of computer vision tasks. An excellent autoencoder-based method [8] has already solved cloud removal focused on using multitemporal images. We therefore also consider using an autoencoder to solve the bothersome problem of cloud cover.

In this letter, we simplify cloud removal by using a single cloud-contaminated image as a combination of image inpainting and denoising problems. We propose Cloud-Aware Generative Network (CAGN) for cloud removal, as shown in Fig. 1. Different from the previous method described in [8], the CAGN uses a single degraded image for restoration without multitemporal information. The pipeline of the network has two stages: the first stage is for cloud region detection and the second stage is for reconstruction. In the first stage, we use a recurrent network to produce a binary mask of the cloud region. We simply consider the region with thick cloud as having lost all information, and the region with thin cloud

Manuscript received March 14, 2019; revised April 18, 2019 and June 11, 2019; accepted July 7, 2019. This work was supported in part by the Youth Innovation Promotion Association, CAS, under Grant 2016201 and in part by the National Defense Science and Technology Innovation Zone under Grant 18-H863-00-TS-002-018-01. *(Corresponding author: Ye Zhang.)*

L. Sun is with the Image Processing Department, Key Laboratory of Airborne Optical Imaging and Measurement, Chinese Academy of Sciences, Changchun Institute of Optics, Fine Mechanics and Physics, Chinese Academy of Sciences, Changchun 130033, China, and also with the College of Materials Sciences and Opto-Electronic Technology, University of Chinese Academy of Sciences, Beijing 100059, China (e-mail: sunlinjian16@mails.ucas.edu.cn).

Y. Zhang, X. Chang, and J. Xu are with the State Key Laboratory of Applied Optics, Changchun Institute of Optics, Fine Mechanics and Physics, Chinese Academy of Sciences, Changchun 130033, China, and also with the AI Lab, Changchun Spirits AI Technology Co., Ltd., Changchun 130033, China (e-mail: zhangye@ciomp.ac.cn; xuling.chang@ciomp.ac.cn; jiajia.xu@ciomp.ac.cn).

Y. Wang is with the Image Processing Department, Key Laboratory of Airborne Optical Imaging and Measurement, Chinese Academy of Sciences, Changchun Institute of Optics, Fine Mechanics and Physics, Chinese Academy of Sciences, Changchun 130033, China (e-mail: wangyj@ciomp.ac.cn).

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

Digital Object Identifier 10.1109/LGRS.2019.2928840

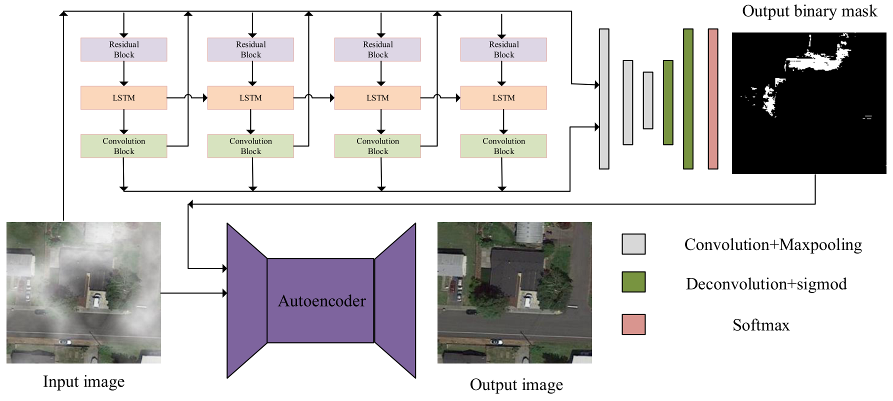

Fig. 1. Architecture of CAGN consists of two subnetworks. The first stage of the network consists of four convolutional LSTM units and a segmentation network to generate a mask of the cloud region. The second stage of the network consists of an autoencoder with partial convolutional layers.

is considered to retain more information about the scene of the ground. We introduce a vision attention mechanism in the first stage, which pays more attention to the region of thick cloud. The second stage is an autoencoder embedded with partial convolutional layers, taking both the raw image and the binary mask as the input to produce a cloud-free image. We use the partial convolutional layers and condition the autoencoder dynamically to complete the region of thick cloud while denoising the region of thin cloud. In addition, we train the CAGN using a synthetic data set. Finally, we fine-tune on a smaller real cloud data set and achieve good results.

The main contributions of this letter are as follows.

1) We propose CAGN, which injects a visual attention mechanism to detect and remove clouds by using a single optical remote sensing image without multitemporal information.
2) We introduce partial convolutional layers embedded in the autoencoder to dynamically condition the thick cloud region to participate in convolution calculations to mitigate blurring.

II. Study Area and Materials

We focus on mitigating the degradation of optical remote sensing images caused by cloud occlusion. We present a method that can capture the region where the land is occluded by clouds and reconstruct the region automatically. For this special task, we need paired training data consisting of a clear remote sensing image and its corresponding degraded image. The influence of clouds on radiation transmission is a complex physical process, which is difficult to simulate completely and realistically. For simplicity, this letter mainly tries to alleviate the problem of cloud occlusion from the perspective of image processing methods. To simulate the near-earth optical remote sensing image degraded by cloud, we make a small-scale synthetic data set using 361 clear satellite images (800×800 pixels) acquired from Google Earth. To obtain the degraded image paired with the clear image, we used Adobe Photoshop to create a transparent layer and then randomly generated the cloud on the transparent layer by using its own rendering function. Finally, stacking the transparent layer with cloud on the clear image, we obtain the degraded image. We crop the original-sized image to 256×256 pixels using the step of 100 pixels along both the horizontal and vertical directions. Finally, we obtain the data set, which consists of 12 996 pairs of optical remote sensing images. The data set was divided into 12 000 image pairs for training and 996 for testing.

The data set for the experiments mainly consists of a triple type. We divide the problem of removing the cloud from optic remote sensing images into two subproblems. The first subproblem is cloud region detection, which can be transferred to coarse-grained binary semantic segmentation. The second problem is the region of image reconstruction, which can be considered as an image inpainting problem. We propose our two-stage model to solve the above two problems. The mask used in training the first-stage model is a binary representation of which pixels belong to the region occluded by cloud. If a pixel belongs to the region of thick cloud, the pixel value is 1; otherwise, its value is 0. During the second-stage training, the mask value is opposite to the mask used during the first stage.

III. Methodology

A. CAGN Architecture Overview

The architecture of the CAGN is shown in Fig. 1. The CAGN consists of two subnetworks, the attention-based cloud region detection subnetwork and the autoencoder subnetwork. In the attention-based cloud region detection subnetwork, we utilize a recurrent network used in [9] to produce an attention map of the cloud region in the input image. Each block of the recurrent network consists of a ResNet with several layers as the feature extractor, a convolutional long short-term memory (LSTM) unit [10], and a convolutional block for generating attention maps. The attention map is learned in multisteps and is a one-channel 2-D matrix of the score. The score ranges from 0 to 1. The greater the value of the score is at a location, the greater the attention it suggests. The attention map represents the increasing possibility from the region not occluded by the cloud to the region that is occluded by the cloud. The LSTM unit plays an important role in the attention map production process. It is a popular architecture of recurrent neural networks. Each LSTM unit consists of a memory unit c, a hidden state h, and three types of gates: the input gate i, the forget gate f, and the output gate o. These units are used to control the read-write memory unit. For each time step t, the LSTM unit first receives input x_t and previous hidden state h_{t-1} and then computes activations of the gates and updates c_t and h_t. The equations involved are shown in (1), where $\odot$ denotes the Hadamard product and $*$ denotes the convolution operation

$$\begin{aligned} i_t &= \sigma(W_{\mathrm{xt}} * x_t + W_{\mathrm{hi}} * h_{t-1} + W_{\mathrm{ci}} \odot c_{t-1} + b_i) \\ f_t &= \sigma(W_{\mathrm{xt}} * x_t + W_{\mathrm{hf}} * h_{t-1} + W_{\mathrm{cf}} \odot c_{t-1} + b_f) \\ c_t &= f_t \odot c_{t-1} + i_t \odot \tanh(W_{\mathrm{xc}} * x_t + W_{\mathrm{hc}} * h_{t-1} + b_c) \\ o_t &= \sigma(W_{\mathrm{xo}} * x_t + W_{\mathrm{hf}} * h_{t-1} + W_{\mathrm{co}} \odot c_t + b_o) \\ h_t &= o_t \odot \tanh(c_t) \end{aligned} \tag{1}$$

where x_t is the feature map extracted by the residual block. c_t acts as an accumulator of the state information that will propagate to the next LSTM unit. h_t represents the output feature of the LSTM unit that will be fed to the convolution layer for generating the attention map. We concatenate the current attention map with the input image in each time step during the forward procedure. The attention map generated by all four times step will be fed to the segmentation network for producing a 2-D binary mask for the cloud region. Finally, we stack the 2-D binary mask to three channels and feed it into the autoencoder.

In the second stage of the CAGN, we use a UNet-like architecture [11] as the autoencoder. Partial convolution has been shown to improve the quality of image reconstruction. We adopt the network design described in [12], which is slightly changed relative to the original network. We replace all batch normalization [13] with instance normalization [14] between each convolution layer and rectified linear unit (ReLU)/LeakyReLU layer except the first and last partial convolutional layers. In the scenario of this letter, we use partial convolutions where the convolution is masked and renormalized to be conditioned on only the region without cloud, as mentioned in Section II. We invert the binary mask produced by the first stage network. Contrary to the procedure producing the attention map, we set the pixel value of the region occluded by the cloud to 0, and we set it to 1 otherwise. The partial convolution at every location, defined in [15], is expressed as follows:

$$x' = \begin{cases} W_T(X \odot M)\frac{1}{\mathrm{sum}(M)} & \text{if } \mathrm{sum}(M) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where W represents the convolution filter weights and b is the corresponding bias. X is the feature map for the current sliding window and M is the corresponding binary mask. After each partial convolution, we update the mask following the rule as follows:

$$m' = \begin{cases} 1 & \text{if } \mathrm{sum}(M) > 0 \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

If the convolution was able to condition its output on at least one valid input value, then we remove the mask for the corresponding location. We ensemble the first-stage and second-stage networks as the whole model, training the two parts of the model separately.

B. Loss Function

We train the first-stage and second-stage networks separately. The loss of the first stage can be formulated as follows:

$$L_{\mathrm{1st}} = L_{\mathrm{att}} + \lambda_{\mathrm{seg}} L_{\mathrm{seg}} \tag{4}$$

where L_{att} and L_{seg} represent the losses for the attention map and the segmentation mask, respectively, and λ_{seg} weighs the importance between two losses. We set λ_{seg} to 10 in training the first-stage network. We use pairs of images with cloud and its cloud region mask. The loss in each step of the convolutional LSTM is defined as the mean squared error (MSE) between A_t, which presents the output attention map in step t, and the binary mask M. We set this process in four steps, initializing the attention map with the values of 0.5. The region without cloud in the attention map becomes smaller as the step t increases, and simultaneously, the region with cloud will become larger. The loss function for the attention map is given as follows:

$$L_{\mathrm{att}} = \sum_{t=1}^{4} \theta_t^{N-t} L_{\mathrm{MSE}}(A_t, M). \tag{5}$$

We set the value of Î, the same as that in [16] to 0.8. For the segmentation loss, we use class-balance cross-entropy introduced in [16], which is given as follows:

$$L_{\mathrm{seg}} = -\beta Y^* \log \hat{Y} - (1-\beta)(1 - Y^* \log(1-\hat{Y})) \tag{6}$$

where $\hat{Y}$ is the prediction of binary mask and $Y^* = M$ is the ground truth. The parameter β is the balancing weight between the positive and negative samples, formulated as follows:

$$\beta = 1 - \frac{\sum_{y^* \in Y^*} y^*}{|Y^*|}. \tag{7}$$

We find that the combination of attention loss and segmentation loss works well for predicting the binary mask. In the scenario of this letter, we require only coarse-gained segmentation results. For convenience, we introduce the attention mechanism as auxiliary.

When training the second-stage network, we need an input image X_{in}, a binary mask M, an output image X_{out}, and the ground truth X_{gt}. Similar to the loss for the task of image inpainting used in [19], we also facilitate pixel losses defined as follows:

$$\begin{aligned} L_{\mathrm{cloud}} &= ||(1-M) \odot (X_{\mathrm{out}} - X_{\mathrm{gt}})||_1 \\ L_{\mathrm{noncloud}} &= ||M \odot (X_{\mathrm{out}} - X_{\mathrm{gt}})||_1 \end{aligned} \tag{8}$$

where $|| \bullet ||_1$ represents L_1 loss. We also use perceptual loss [17] in multiscale defined as follows:

$$\begin{aligned} L_{\mathrm{perceptual}} = &\sum_{n=0}^{N-1} ||\mathrm{VGG}_n(X_{\mathrm{out}}) - \mathrm{VGG}_n(X_{\mathrm{gt}})||_1 \\ &+ \sum_{n=0}^{N-1} ||\mathrm{VGG}_n(X_{\mathrm{comp}}) - \mathrm{VGG}_n(X_{\mathrm{gt}})||_1 \end{aligned} \tag{9}$$

where $X_{\mathrm{comp}} = (1-M) \odot X_{\mathrm{out}} + M \odot X_{\mathrm{gt}}$. The perceptual loss computes the L_1 distance of features after projecting these images into higher feature space. We use a VGG16 network [17] pretrained on the ImageNet data set as the feature extractor, and VGG_n is the output layer that we select. We use the output of pooling layers from 1 to 3. Next, we introduce the style loss, which is usually used in style transfer [18] and image reconstruction. We compute the Gram matrix using the same feature maps as perceptual loss, defined as follows:

$$L_{\text{style}} = \sum_{n=0}^{N-1} \left|\left| K_n \left({F_{\text{out}}^n}^T F_{\text{out}}^n - {F_{\text{gt}}^n}^T F_{\text{gt}}^n \right) \right|\right|_1 + \sum_{n=0}^{N-1} \left|\left| K_n \left({F_{\text{comp}}^n}^T F_{\text{comp}}^n - {F_{\text{gt}}^n}^T F_{\text{gt}}^n \right) \right|\right|_1 \quad (10)$$

where $F_{\text{out}}^n = \mathrm{VGG}_n(X_{\text{out}})$, $F_{\text{comp}}^n = \mathrm{VGG}_n(X_{\text{comp}})$, and $F_{\text{gt}}^n = \mathrm{VGG}_n(X_{\text{gt}})$. The $k_n = 1/C_n H_n W_n$ is the normalization factor for the corresponding layer. We add a mean square term for all the pixels of the image, which is defined as follows:

$$L_{\text{MSE}-\text{pixel}} = L_{\text{MSE}}(X_{\text{out}}, X_{\text{gt}}). \quad (11)$$

The last term of the loss for the second-stage network is the total variation loss defined as follows:

$$L_{\text{tv}} = \sum_{(i,j)\in C,(i,j+1)\in C} \left|\left| X_{\text{comp}}^{i,j+1} - X_{\text{comp}}^{i,j} \right|\right|_1 + \sum_{(i,j)\in C,(i+1,j)\in C} \left|\left| X_{\text{comp}}^{i+1,j} - X_{\text{comp}}^{i,j} \right|\right|_1 \quad (12)$$

where C is one pixel belonging to the region of the cloud. It is the smooth penalty term on the region of cloud. The loss of the second-stage network is defined as follows:

$$L_{2\text{st}} = 10L_{\text{cloud}} + 10L_{\text{noncloud}} + 10L_{\text{MSE}-\text{pixel}} + 0.5L_{\text{perceptual}} + 120L\text{style} + L_{\text{tv}}. \quad (13)$$

The weight for each term of the loss was determined by the result observed during training.

IV. Experiment

A. Training Detail

We implement the CAGN with PyTorch and train the model on an NVIDIA GeForce 1060 GPU. Both the first and second stages are trained using the ADAM [19] optimizer. We set the mini-batch to 4 for the first-stage network training procedure; the learning rate of ADAM starts from 1e-2, decays to one-tenth every 20 000 iterations, and stops at 1e-4. We set the mini-batch to 1 for the second-stage network training procedure; the learning rate of ADAM starts from 2e-3, decays to 0.25 every 24 000 iterations, and stops at 96 000 iterations. Finally, we fine-tune the second-stage network, freezing all the instance normalization layers, and use the learning rate 5e-5 by training 24 000 iterations. Both the first-stage network and the second-stage network are trained until performance stops improving. The choice of the learning parameters for both the stages is based on heuristic search.

B. Results

The result of the CAGN is shown in Fig. 2. We select six images [see Fig. 2(a)–(f)] with different scenes and clouds for illustration. The comparisons between the output image and the ground truth image have shown that the CAGN can generate results more similar to the ground truth. We select pix2pix [20] as the baseline model and select the peak signal-to-noise ratio (PSNR) and the structural similarity index (SSIM) as metrics for cloud removal, and we make comparisons between the input images and the output images. Tables I and II show the results of PSNR and SSIM for six images. The CAGN can achieve a considerable increase in PSNR and SSIM of 13.877% and 16.266% on average, respectively, compared with the baseline on synthetic test data. It can be seen from Fig. 2(a)–(e) that the removal effect of thin cloud is very obvious. The thick cloud removal effect can be seen in Fig. 2(f). The pix2pix is also an autoencoder-based method for image reconstruction with adversarial training. The CAGN can provide a better result than pix2pix due to the partial convolutional layer embedded in the second-stage subnetwork. The approach can inhibit the convolution of the pixels belonging to the thick cloud and gradually recover the region from the edge to center of the thick cloud region. We test both the CAGN and pix2pix on an NVIDIA GeForce 1060 GPU. The size of the input-degraded image in the test is the same as that in the training. The time consumption for both the models was 187 ms for the CAGN and 133 ms

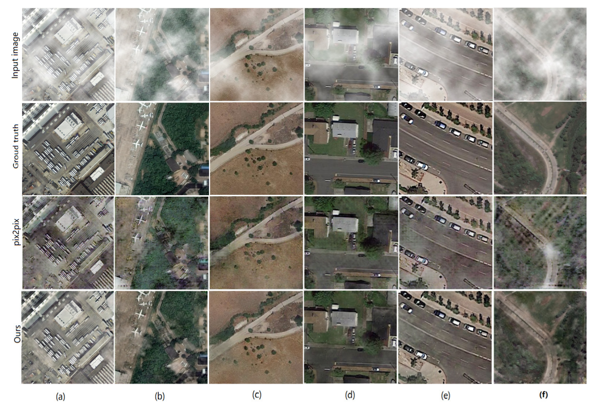

Fig. 2. Result of the CAGN. (a)–(f) We select six images with different scenes and clouds, including both thin cloud and thick cloud for illustration.

TABLE I
PSNR of Cloud Removal

Figure	Input Image	Pix2Pix [20]	CAGN
(a)	10.642917	17.109872	17.703568
(b)	9.428533	17.140081	19.532626
(c)	14.219789	22.400406	27.023368
(d)	10.679250	21.017073	23.706016
(e)	10.143456	23.110806	24.907590
(f)	9.054431	19.110574	21.179471

TABLE II
SSIM of Cloud Removal

Figure	Input Image	Pix2Pix [20]	CAGN
(a)	0.684391	0.673837	0.784343
(b)	0.597525	0.571707	0.769919
(c)	0.842942	0.752108	0.891969
(d)	0.689876	0.783567	0.858410
(e)	0.708236	0.774919	0.880334
(f)	0.591352	0.460703	0.736411

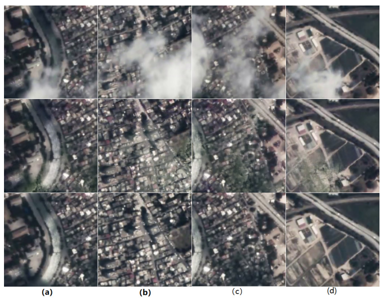

Fig. 3. Removal result of CAGN on real cloud images. (a)–(d) We select four images to illustrate. Each row from top to bottom corresponds to the cloud image, the result of CAGN, and ground truth.

TABLE III
RESULT OF CAGN FOR REAL CLOUD IMAGES

	PSNR		SSIM	
Figure	Input Image	CAGN	Input Image	CAGN
(a)	16.725899	25.426206	0.712899	0.870088
(b)	11.711759	21.889006	0.263901	0.799115
(c)	13.740002	23.541139	0.453740	0.827106
(d)	17.753695	25.662729	0.699081	0.848912

for pix2pix. The CAGN takes 48 ms to produce the mask and 139 ms to reconstruct, and pix2pix has no component to produce a mask. We also collected a set of remote sensing images of Turkey from the JILIN-I satellite on May 25, 2017. We cropped 324 patches of 256×256 pixels. We divided the training set and the test set according to the ratio of 4:1. We fine-tuned CAGN on a real cloud image data set using a pretrained model on the synthetic data set. As shown in Fig. 3, we selected four images to illustrate the test results. The corresponding quantified results are shown in Table III. We think that it can be improved by training on larger scale real data sets in future work.

V. CONCLUSION

In this letter, CAGN has been proposed for removing clouds from optical remote sensing images without multitemporal information. The CAGN consists of a two-stage model with a first-stage network for detecting the region of the cloud and the second-stage network for cloud removal. We introduce an attention mechanism for generating the binary mask for cloud regions in the first-stage network, and we use partial convolution layers to condition the region of the completed image. We also produce a small synthetic data set to train the CAGN and fine-tune using real images with cloud occlusion. The result shows that the CAGN can achieve significant effects.

REFERENCES

[1] C. H. Lin, P. H. Tsai, K. H. Lai, and J. Y. Chen, "Cloud removal from multitemporal satellite images using information cloning," *IEEE Trans. Geosci. Remote Sens.*, vol. 51, no. 1, pp. 232–241, Jan. 2013.

[2] Z. Zhu and C. E. Woodcock, "Object-based cloud and cloud shadow detection in Landsat imagery," *Remote Sens. Environ.*, vol. 118, pp. 83–94, Mar. 2012.

[3] A. Maalouf, P. Carre, B. Augereau, and C. Fernandez-Maloigne, "A bandelet-based inpainting technique for clouds removal from remotely sensed images," *IEEE Trans. Geosci. Remote Sens.*, vol. 47, no. 7, pp. 2363–2371, Jul. 2009.

[4] L. Lorenzi, F. Melgani, and G. Mercier, "Inpainting strategies for reconstruction of missing data in VHR images," *IEEE Geosci. Remote Sens. Lett.*, vol. 8, no. 5, pp. 914–918, Sep. 2011.

[5] H. Li, L. Zhang, H. Shen, and P. Li, "A variational gradient-based fusion method for visible and SWIR imagery," *Photogramm. Eng. Remote Sens.*, vol. 78, no. 9, pp. 947–958, Sep. 2012.

[6] H. Shen, H. Li, Y. Qian, L. Zhang, and Q. Yuan, "An effective thin cloud removal procedure for visible remote sensing images," *J. Photogramm. Remote Sens.*, vol. 96, pp. 224–235, Oct. 2014.

[7] M. Xu, X. Jia, M. Pickering, and A. J. Plaza, "Cloud removal based on sparse representation via multitemporal dictionary learning," *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 5, pp. 2998–3006, May 2016.

[8] S. Malek, F. Melgani, Y. Bazi, and N. Alajlan, "Reconstructing cloud-contaminated multispectral images with contextualized autoencoder neural networks," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 4, pp. 2270–2282, Apr. 2018.

[9] R. Qian, R. T. Tan, W. Yang, J. Su, and J. Liu, "Attentive generative adversarial network for raindrop removal from a single image," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2018, pp. 2482–2491.

[10] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-C. WOO, "Convolutional LSTM network: A machine learning approach for precipitation nowcasting," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 802–810.

[11] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Proc. Int. Conf. Med. Image Comput. Comput.-Assist. Intervent.*, Nov. 2015, pp. 234–241.

[12] G. Liu, F. A. Reda, K. J. Shih, T.-C. Wang, A. Tao, and B. Catanzaro, "Image inpainting for irregular holes using partial convolutions," Dec. 2018, *arXiv:1804.07723*. [Online]. Available: https://arxiv.org/abs/1804.07723

[13] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," Mar. 2015, *arXiv:1502.03167*. [Online]. Available: https://arxiv.org/abs/1502.03167

[14] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Instance normalization: The missing ingredient for fast stylization," 2016, *arXiv:1607.08022*. [Online]. Available: https://arxiv.org/abs/1607.08022

[15] S. Xie and Z. Tu, "Holistically-nested edge detection," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1395–1403.

[16] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *Proc. Eur. Conf. Comput. Vis.*, Sep. 2016, pp. 694–711.

[17] K. Simonyan and A. Zisserman, "'Very deep convolutional networks for large-scale image recognition," Apr. 2014, *arXiv:1409.1556*. [Online]. Available: https://arxiv.org/abs/1409.1556

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

[19] 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

[20] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1125–1134.