Manipulation and generation of synthetic satellite images using deep learning models

Lydia Abady[a,*,†], **János Horváth**[b,*,†], **Benedetta Tondi**[a], **Edward J. Delp**[b] and **Mauro Barni**[a]

[a]University of Siena, Department of Information Engineering and Mathematics, Siena, Italy
[b]Purdue University, School of Electrical and Computer Engineering, West Lafayette, Indiana, United States

Abstract. Generation and manipulation of digital images based on deep learning (DL) are receiving increasing attention for both benign and malevolent uses. As the importance of satellite imagery is increasing, DL has started being used also for the generation of synthetic satellite images. However, the direct use of techniques developed for computer vision applications is not possible, due to the different nature of satellite images. The goal of our work is to describe a number of methods to generate manipulated and synthetic satellite images. To be specific, we focus on two different types of manipulations: full image modification and local splicing. In the former case, we rely on generative adversarial networks commonly used for style transfer applications, adapting them to implement two different kinds of transfer: (i) land cover transfer, aiming at modifying the image content from vegetation to barren and vice versa and (ii) season transfer, aiming at modifying the image content from winter to summer and vice versa. With regard to local splicing, we present two different architectures. The first one uses image generative pretrained transformer and is trained on pixel sequences in order to predict pixels in semantically consistent regions identified using watershed segmentation. The second technique uses a vision transformer operating on image patches rather than on a pixel by pixel basis. We use the trained vision transformer to generate synthetic image segments and splice them into a selected region of the to-be-manipulated image. All the proposed methods generate highly realistic, synthetic, and satellite images. Among the possible applications of the proposed techniques, we mention the generation of proper datasets for the evaluation and training of tools for the analysis of satellite images. [DOI: 10.1117/1.JRS.16.046504]

Keywords: manipulated remote sensing images; generation of synthetic satellite images; style-transfer generative adversarial networks; Sentinel-2; image generative pretrained transformer; vision transformer.

Paper 220106G received Feb. 25, 2022; accepted for publication Oct. 17, 2022; published online Nov. 1, 2022.

1 Introduction

Manipulation and generation of synthetic images by means of deep learning (DL) architectures are receiving increasing attention due to the demand of large labeled datasets for artificial intelligence (AI) applications.[1,2] The usage of synthetically generated images for the entertainment industry and even for malevolent disinformation campaigns is also growing. Moreover, satellite images are receiving increasing attention in several application areas including meteorological forecasts and the monitoring and detection of natural disasters. As a result, the number of commercial satellites is constantly growing and the accessibility of imagery is increasing on a daily basis. With the application of AI tools to the analysis and processing of satellite images and the consequent necessity of gathering adequate amounts of labeled data for training, it is natural that the quest for tools capable of generating synthetic data also applies to this kind of image.[3–5]

*Address all correspondence to Lydia Abady, abady@diism.unisi.it; János Horváth, horvath5@purdue.edu
†These authors equally contributed in this work.

The malevolent uses of synthetic and manipulated satellite images are also possible. The development of tools for the detection and localization of manipulated satellite images[6–11] also requires the availability of adequate training and test datasets. The goal of this paper is to describe a number of methods to generate globally and locally manipulated satellite images.

While deep neural networks have been successfully applied to the generation and tampering of natural images and multimedia content, their use to generate synthetic satellite images has received limited attention. In addition, the tools developed for media applications cannot be used directly to generate synthetic satellite images, due to the different nature of such images in terms of semantic content, number of bands, bit, and spatial resolution. In this paper, we present a number of DL architectures for the generation of synthetic and manipulated satellite images, focusing on two different kinds of manipulations: full image modification and local splicing. We demonstrate the validity of the proposed methods using Sentinel-2 images.[12]

Generative adversarial networks (GANs) are popular DL architectures widely used for both synthetic image generation[13] and image style transfer.[14] In this paper, we use them for the global manipulation of satellite images. In particular, we use them for transferring the style of satellite images in such a way to change their overall content and semantic meaning. In contrast, we do not report any efforts to generate satellite images from scratch (as done, for example, in Ref. 15). To be specific, we apply two different kinds of image style transfer. The first one, referred to as land cover transfer, aims at changing the land cover of the manipulated images from vegetation to barren and vice versa. To do so, we rely on a properly trained version of the cycleGAN architecture.[16] The goal of the second global transformation is season transfer, whereby summer satellite images are transformed into winter ones and vice versa. For such transformation, we use the pix2pix GAN architecture.[14] The final goal of these transformations is to obtain fully synthetic images that can be used to construct large labeled datasets.

With regard to local tampering, we present two types of transformer-based image generation techniques.[17] In particular, we generate the manipulated images by inserting synthetic splices into irregular regions of the target image, making sure that the spliced boundaries are not visible. Transformers were initially used for natural language processing applications.[18] However, in the last few years, they were also used to process still images.[17,19] The first architecture we present is the image generative pretrained transformer (iGPT),[19] which is an autoregressive network trained to predict pixels without observing the entire region of the 2D input image. The iGPT is trained to generate the missing part of a satellite image from its neighboring pixels. In this way, parts of the image can be removed without introducing noticeable artifacts. The second method is based on a vision transformer, originally developed for image classification.[17] We use the vision transformer to create synthetic regions (splices) of an image by modifying the output of the last layer of the transformer. The final goal of the local image manipulation architectures is the creation of a large labeled dataset of images with synthetically generated splices.

The remainder of this paper is organized as follows. In Sec. 2, we provide a brief review of the state-of-the art of satellite images generation and manipulation using DL networks. In Sec. 3, we list the datasets used throughout our work. Then in Sec. 4, we describe the architectures for the generation of synthetic manipulated images. In Sec. 5, we provide some examples of the images produced by the proposed architectures. Finally, in Sec. 6, we summarize the results of our work and make some final remarks.

2 Relevant Work

GAN architectures have been successfully used to generate face images with an extremely high level of realism.[20–22] Lately, they have also been used to generate synthetic images with nonfacial content.[23] Even more recently, these techniques have started being used to generate synthetic remote sensing images. In Ref. 15, a progressive GAN[20] was trained on all 180k samples of SEN12MS dataset[24] to generate 13-band$_s$ images that imitate Sentinel-2 level-1C products.[25] GANs have also been used to generate different types of satellite images that are correlated with the input images. For example, Fuentes Reyes et al.[3] used a GAN to generate optical images starting from synthetic aperture radar (SAR) images. They did so by training a cycleGAN architecture with 512×512 patches of SAR images as input (domain A) and optical images as a

reference (domain B). They constrained the output of the network to be a gray-scale image similar to SAR images. In addition to synthesizing fake images from scratch and generating different types of data, several papers have used GANs for image quality improvement, noticeably cloud removal,[26–28] and sharpening.[29–32] A pix2pix architecture was used in Ref. 5 to generate satellite images starting from 256×256 historical maps and RGB optical satellite reference images.

Despite the increasing interest in the use of GANs for satellite images, only a few works have used GANs to change the semantic content of existing images, which is the goal of this paper. To the best of our knowledge, the only papers dealing with this issue are Refs. 15, 33, and 34. Abady et al.[15] proposed an image-to-image translation approach to change the land-cover of a satellite image. Specifically, a NICE GAN[35] is applied to achieve land cover transfer on four-band [RGB and near-infrared (NIR) bands] images of 480×480 resolution. The land cover transfer regards the transformation of vegetation terrains to barren and vice versa. A similar task is addressed in Ref. 33 where the authors exploit a CycleGAN architecture to translate 10 bands of a Sentinel-2 level-1C image, namely the 10- and 20-m bands, from barren to vegetation and vice versa. Finally, in Ref. 34, a method is presented for the creation of synthetic images having the urban structure of a given city (i.e., Tacoma in Washington, USA) but with the landscape features of another city (i.e., Seattle in Washington, USA and Beijing, China). A cycleGAN architecture is used to transfer the style of cartoDB basemap to satellite images and vice versa. The cycleGAN model is trained on a specific city B to generate simulated satellite images from the basemap of another city A.

The techniques for global manipulations proposed in this paper focus on land-cover and season transfer. With regard to land cover transfer, the method we propose is a highly improved version of the technique described in Ref. 15. The new approach is based on CycleGAN instead of NICE GAN and produces transferred images with very good quality in both directions (vegetation to barren and vice versa). In Ref. 15, instead, the transfer was successful only in one direction (vegetation to barren), whereas in the other direction (barren to vegetation,) the quality of the transferred images was poor.[15] As to season transfer, this is a new kind of manipulation that has never been addressed before. In particular, we propose an architecture whose application transforms a satellite image taken in the winter (summer) into its summer (winter) counterpart.

With regard to local splicing, to the best of our knowledge, no work has been proposed in the literature performing local tampering of remote sensing images using generative models. Therefore, our paper represents a first attempt in this direction.

3 Datasets

In this section, we describe the datasets we have used in our experiments to demonstrate the validity of the techniques we have developed. The datasets have been used to train the architectures proposed in this paper and to assess their performance. All the datasets are obtained starting from Sentinel-2 products;[12] however, our methods can also be used on imagery from other satellites, we have chosen Sentinel-2 images because of their availability for research goals.

For global manipulations, we have used Sentinel-2 level-1C images, whereas for local manipulations, we have used both Sentinel-2 level-1C images and Sentinel-2 level-2A images. Sentinel-2 level-1C images consist of 13 bands, with band 2 representing the green channel, band 3 the blue channel, and band 4 the red channel. Band 8 is one of the NIR channels. Bands 2 (green), 3 (blue), 4 (red), and 8 (NIR) have a spatial resolution of 10 m, with size $10{,}980 \times 10{,}980$. Other six bands (bands 5, 6, 7, 8a, 11, and 12) have a spatial resolution of 20 m, for a size of 5490×5490 pixels. Finally, bands 1, 9, and 10 have a spatial resolution of 60 m, with size equal to 1830×1830. Sentinel-2 level-2A is the bottom-of-atmosphere product obtained by applying atmospheric correction to top-of-atmosphere level-1C products. Its bands are similar to those of the level-1C products, except for band 10, which is not present. All the Sentinel-2 level-1C images were downloaded directly from the ESA Copernicus hub,[36] whereas Sentinel-2 level-2A images were taken from the dataset SEN12MS.[24] In Table 1, we summarize the datasets that we have used in our research.

Table 1 Datasets used for our experiments.

Dataset	Image size	Radiometric resolution (bits)	Bands	Dataset size	Task
Alps	512×512	16	4	3936 pairs	Season transfer
Scandinavia	512×512	16	4	9000 pairs	Season transfer
Land cover	512×512	16	4	20,000	Land cover transfer
SEN12MS	256×256	16	3	120,000	iGPT and vision transformer training
World	512×512	16	3	285,768	iGPT and vision transformer splice insertion

3.1 *Alps Dataset*

We designed the Alps dataset to train and test the architecture for the season transfer manipulation. The alpine area, in fact, is characterized by marked differences between winter and summer, with winter images largely covered by snow, and summer images containing large areas of green vegetation. For this dataset, we only selected the RGB and NIR bands, with ground sampling distance (GSD) equal to 10 m, for a total size of $10{,}980 \times 10{,}980$ resolution and 16 bits per pixel. We collected images representing exactly the same area taken at two different months, each month representing a different season. In this way, in addition to using the images for training the season transfer architecture, we also have a way to compare the results of the season transfer with ground-truth images. In particular, we selected images taken in June 2019 for the summer dataset and in December 2019 for the winter dataset.

The description of the procedure we followed to build the dataset is described in the following. To start, we selected only images with limited cloud coverage. Since it was not possible to get only images with 0% cloud coverage, we limited the search to images with a cloud coverage lower than 9%. We extracted the bands of interest (RGB and NIR) from the downloaded products as jp2000 images. We used the gdal software library[37] for reading and writing raster and vector geospatial data formats. Specifically, we used the gdal retile command to tile the downloaded images from their initial size into several 512×512 images. Then we removed the edge tiles. Finally, we paired the images of areas existing in both the winter and summer domains. As a result, we built a dataset of 3936 pairs of images with 512×512 resolution. In Fig. 1, we show the RGB version of some sample images of the Alps dataset.

3.2 *Scandinavian Dataset*

To validate the effectiveness of the season transfer architecture on different kinds of landscapes, we built another dataset with marked differences between winter and summer. The dataset includes images from Scandinavia and was built similarly to the Alps dataset. In this case, the selected date range includes June 2020 for the summer and February 2020 for the winter. For both domains, we selected only images with 0% cloud coverage. The final dataset consists of 9000 paired images of size 512×512. Figure 1 shows some RGB examples of the images from the Scandinavian dataset.

3.3 *Land Cover Datasets*

The second kind of global manipulation we have considered aims at transforming barren covers into vegetation areas and vice versa. For this reason, we created the land cover dataset by selecting images, in which one of the two types of covers is predominant with respect to the other. To do so, we first selected the areas of interest based on the statistics provided by the organization for economic co-operation and development.[38] With regard to the areas with a high percentage

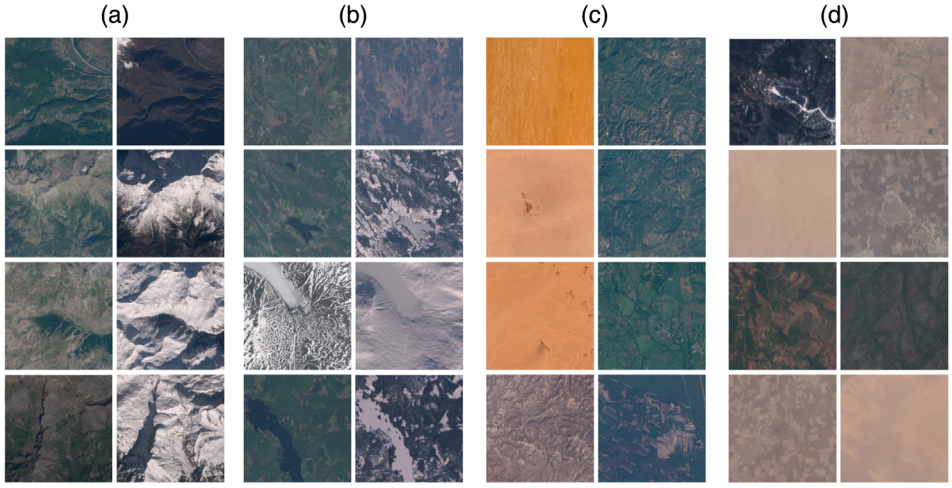

Fig. 1 Example images from the datasets: (a) Alps pairs, (b) Scandinavian pairs, (c) land cover images, and (d) SEN12MS images.

of vegetation, we extracted images from Congo, Salvador, Montenegro, Gabon, and Guyana. The cloud coverage was set to 0%, and the range of dates spanned from June 2019 till December 2019. For the barren domain, we selected the areas of interest in South and Central America. For both domains, we used a linear discriminant analysis classifier[39] to make sure that after cropping the images to 512×512 patches, they contain, respectively, a great percentage of vegetation and barren soil. For each domain, we collected 10,000 images. For the vegetation domain, the average percentage of vegetation pixels in an image is 98% with a maximum of 100% and a minimum of 60%. For the barren domain, the average percentage of barren pixels in an image is 82% with a maximum of 100% and a minimum of 63%. In Fig. 1, we show some RGB examples for the two different domains.

3.4 *SEN12MS*

The SEN12MS[24] dataset was created to provide a large-scale satellite dataset for developing DL-based methods. As opposed to the previous datasets, SEN12MS has a larger variety of images regarding spatial coverage, diversity, and number of available samples. The dataset contains 180,662 triplets of multispectral patches, dual-pol SAR image patches, and MODIS land cover maps collected from Sentinel-1 and Sentinel-2 satellite imagery. The images span all seasons with an approximately equal number of images captured during winter, spring, summer, and fall. The images' locations vary with respect to sea level elevation, climate, latitude, and urbanization level. Each patch has a resolution of 256×256 pixels and we only used the RGB channels of the multispectral patches. In our experiments, we used images from Africa, Europe, Asia, Australia, and South America, for a total of $\sim 120,000$ images. Figure 1 shows some RGB images from SEN12MS.

3.5 *World Dataset*

Eventually, we collected images from various regions in the world by downloading them from the Copernicus Hub to construct the world dataset. The images were captured in 2018 and have an equal number of images across seasons. We only used the RGB bands, which are sampled at 10-m GSD with image size equal to $10,980 \times 10,980$. The images can contain clouds up to 2% of the total number of pixels. From these images, we extracted nonoverlapping patches of size 512×512. The world dataset contains 285,768 images. We used the world dataset images to generate images containing spliced objects as described in the subsequent sections. Figure 2 shows four examples of world dataset images.

Fig. 2 Example of images from the world dataset.

4 DNN Models

In this section, we describe the architectures we used to create the synthetic images. We start with the architectures for global manipulations followed by those targeting local manipulations.

4.1 *Season Transfer*

As we said, with regard to global manipulation, we considered two different objectives. The first objective was to transfer images taken in the winter to their summer counterpart and vice versa. Paired images, with synchronized input and ground truth images, are not difficult to get in this case since the same location is usually available for download in both seasons. For this reason, and since using paired images facilitates training, we used the pix2pix architecture.[14] Pix2pix is a variant of traditional GANs, where instead of generating an image from noise, a conditional GAN (cGAN)[40] is considered. The cGAN takes one image as input and translates it into an image belonging to the target domain. The training workflow of the pix2pix is shown in Fig. 3. The generator attempts to generate a winter image (x) starting from its summer counterpart (x') and vice versa. The goal of the discriminator is similar to that of a classical GAN, namely to judge if an image is a real or a synthetic one. As opposed to classical GANs, however, the discriminator takes as input a pair of images, an original image from the source season, and an image depicting the same area in the target season, the goal being to decide if the image of the target season is real or fake. A model can be trained to transfer images in one direction only,

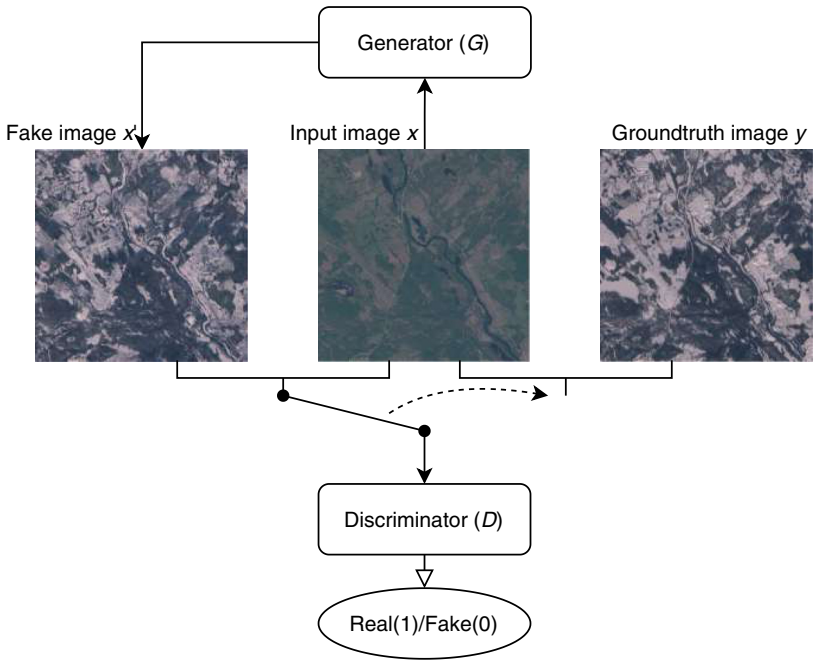

Fig. 3 Pix2pix training workflow.

which is from winter to summer or from summer to winter, so to be able to apply bidirectional transfers, we had to train two models.

4.1.1 *Architecture*

The generator part of the pix2pix season transfer architecture consists of 8 U-Net blocks with skipped connections,[41] and the input size of the first layer is $512 \times 512 \times 4$. Each block is made up of two convolutional layers, two batch normalization layers, a leaky ReLU activation function layer with drop out 0.2, and an ReLU activation function layer with drop out 0.2. As for the discriminator, we used seven convolutional layers, each followed by batch normalization and leaky ReLU activation. The input of the discriminator is the input image x concatenated either with the ground truth image y or the generated image x'. Both the generator and the discriminator have been trained using Adam optimization.[42]

The loss function used to train the pix2pix architecture is made up of two partial losses. The first one is the adversarial binary cross entropy loss L_{adv}:

$$L_{\text{adv}} = E_{x,y}[\log\, D(x, y)] + E_x[\log(1 - D(x, G(x)))], \tag{1}$$

where D represents the discriminator network, G represents the generator network, x is the to-be-translated image, and y is the ground truth image of x in the target season. The second partial loss aims at minimizing the L_1 distance between the generated image ($G(x)$) and its ground truth counterpart (y):

$$L_1 = E_{x,y}[\|y - G(x)\|]. \tag{2}$$

The objective of the generator is to minimize a weighted combination of L_{adv} and L_1, that is:

$$\min_G L_{\text{adv}} + \lambda L_1, \tag{3}$$

where λ should be set to adjust the relative weights of the partial losses. Finally, the discriminator's objective is to distinguish real and synthetic images, that is to maximize L_{adv}.

4.1.2 *Training*

We trained the pix2pix architecture on the Alps and the Scandinavian datasets. Both datasets are characterized by extensive snow coverage in the winter and large green vegetation areas in the summer. The main difference between the datasets is that Alps images are mostly mountainous, whereas the majority of the Scandinavian dataset consists of meadows. For the Scandinavian dataset, 6000 images were used for training, 2000 for testing, and 1000 for validation. Although for the Alps dataset, 2800 images were used for training, 787 images for testing, and 349 images for validation. In total, we trained four models, corresponding to two different transfer directions for each dataset. Training a model took about 4.5 days on one NVIDIA GeForce RTX 2070 with Max-Q Design. To create the season transferred images, the pix2pix model was, separately, trained on the Alps and Scandinavian datasets. The optimization parameters selected for the Adam optimizer were $\beta_1 = 0.5$ and $\beta_2 = 0.999$. We set the learning rate to 0.0001. The number of selected filters is 64, and the slope of the leaky ReLU was set to 0.2. Each model was trained for 200 epochs with a batch size equal to 1. The weight for the L1 loss function λ was set to 100.

4.2 *Land Cover Transfer*

The second global manipulation we have considered is land cover transfer, whereby barren images are transferred to vegetation images and vice versa. A noticeable difference with respect to season transfer is that, in this case, we have no ground truth images since in most cases the barren (res. vegetation) version of a vegetation (res. barren) image does not exist. This requires that a different architecture and training strategy be used. In particular, we opted for the

CycleGAN architecture,[16] which, unlike pix2pix, does not require the availability of paired images for training.

The CycleGAN architecture consists of two generators and two discriminators. Let us assume that the goal of the CycleGAN is to transfer images from a domain A to a domain B and vice versa. Each generator translates the images in one direction. Specifically, the first generator transfers images from domain A to domain B, whereas the second generator transforms the images in the opposite direction. In this way, each generator can act as an additional constraint for the other. The basic idea behind CycleGAN is to enforce a cyclic consistency in such a way that when the output of the first generator is used as an input for the second, the image produced by the second generator should be as close as possible to the original input image (thus avoiding the need for paired images belonging to the two domains). Note that, unlike with pix2pix, it is not necessary to carry out two separate training for the two directions of the transfer since the two generators that are part of cycleGAN architecture provide the models for the two directions.

4.2.1 *Architecture*

The exact architecture we have used to implement the land cover cycleGAN is described in the following. The generators are implemented by means of a residual network[43] with six residual blocks and skip connections. The input size we used is $512 \times 512 \times 4$. Each residual block has a convolutional layer, a batch normalization layer, and a leaky ReLU activation function layer. Regarding the discriminator, we used seven convolutional layers, each followed by batch normalization and a leaking ReLU activation. For both networks, Adam's optimizer was used. Figure 4 shows the different losses of a cycleGAN architecture. Training is obtained by finding a good trade-off between three partial losses. The first partial loss is the typical adversarial GAN cross entropy loss (L_{adv}) defined as

$$L_{\text{adv}} = E_v[(D_b(G_{v2b}(v)) - 1)^2] + E_b[(D_v(G_{b2v}(b)) - 1)^2], \tag{4}$$

where G_{v2b} is the generator that translates the images from vegetation to barren, G_{b2v} is the generator that transfers the images from barren to vegetation, D_b and D_v are the discriminator networks that classify images as real or fake for the barren and vegetation domains, respectively, and b and v are generic barren and vegetation input images. The second loss is the cyclic consistency loss defined by

$$L_{\text{cycle}} = E_v[\|G_{b2v}(G_{v2b}(v)) - v\|] + E_b[\|G_{v2b}(G_{b2v}(b)) - b\|], \tag{5}$$

whose goal is to make sure that vegetation (res. barren) images that are translated to barren (res. vegetation) and then back to vegetation (res. barren) are as close as possible to the original input.

Finally, an extra constraint is added to ensure that when a generator is fed with an image belonging to the output domain, it leaves the image as is since no transformation is actually necessary. Such a goal is achieved by defining a third loss, namely the identity loss, as follows:

$$L_{\text{identity}} = E_v[\|G_{b2v}(v) - v\|] + E_b[\|G_{v2b}(b) - b\|]. \tag{6}$$

The goal of the generators is to minimize an overall loss combining the three partial losses described above:

$$\min_{G_{v2b}, G_{b2v}} \alpha_1 L_{\text{adv}} + \alpha_2 L_{\text{cycle}} + \alpha_3 L_{\text{identity}}, \tag{7}$$

where α_1, α_2, and α_3 are the weights of the adversarial, cycle, and identity losses, respectively. In contrast, the discriminators aim at distinguishing between real and synthetic images, each in its domain, a goal that is achieved by solving the following optimization problem:

$$\max_{D_v, D_b} E_b[(D_b(b) - 1)^2] + E_v[(D_b(G_{b2v}(v)))^2] + E_v[(D_v(v) - 1)^2] + E_b[(D_v(G_{v2b}(b)))^2]. \tag{8}$$

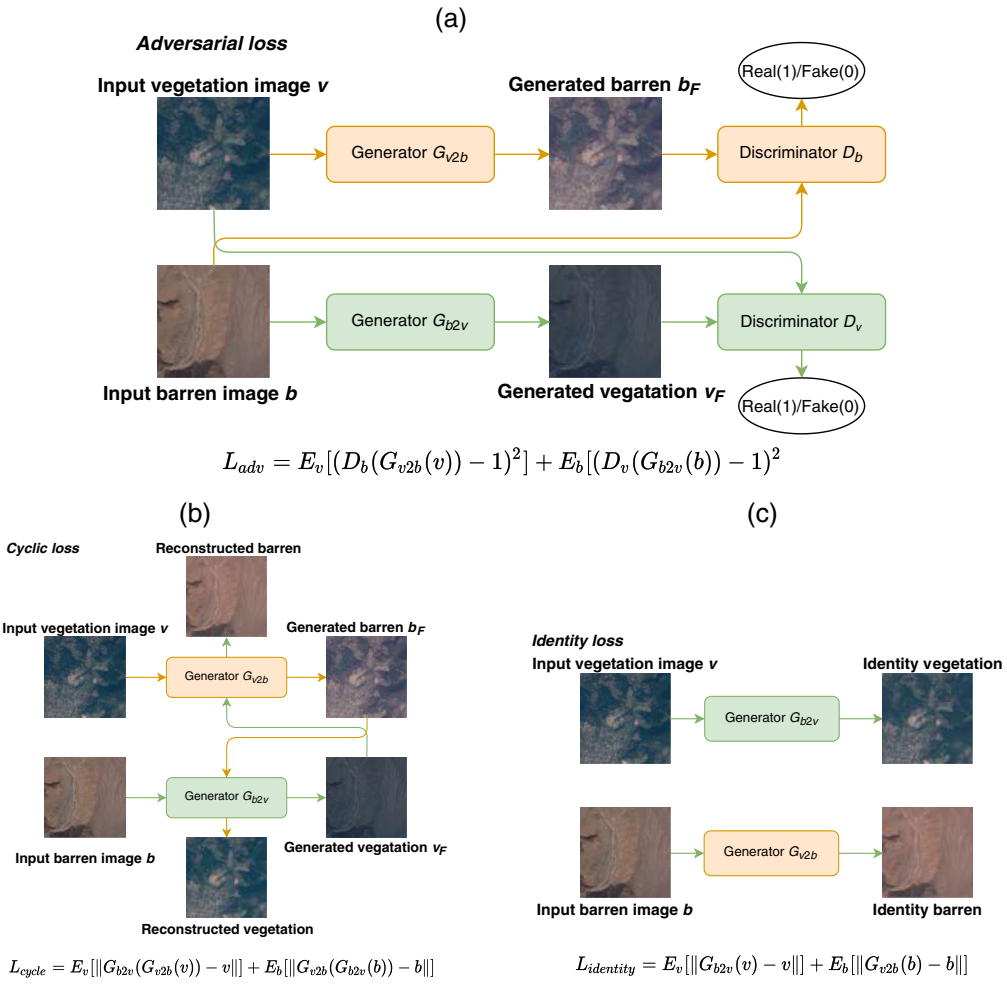

Fig. 4 CycleGAN partial losses: (a) adversarial, (b) cyclic, and (c) identity losses.

4.2.2 *Training*

The dataset we used to train the cycleGAN architecture described in the previous section is the land cover dataset with 8000 images from each domain used for training and 2000 images were kept for testing. Training the model took about 10 days on one NVIDIA GeForce RTX 2070 with Max-Q design GPU. For this task, the cycleGAN model was trained for 200 epochs with an input size of $512 \times 512 \times 4$. For each network of the model, the Adam optimizer was used with β_1 set to 0.5, β_2 to 0.999 and the learning rate set to 0.0001. The number of filters used is 32 and the slope for the leaky ReLU was set to 0.2. The batch size was constrained to 1. The GAN adversarial loss weight α_1 was set to 1 and the cyclic consistency weight α_2 was set to 5, whereas the identity loss weight α_3 was set to 3.

4.3 *Splicing with iGPT*

The next manipulation we considered is local splicing. We used iGPT to generate synthetic image splices that then we inserted into images from the world dataset. The iGPT was trained on the SEN12MS dataset. The goal in this case was to remove some parts of an image from the world dataset and replace them with content generated by iGPT, by paying attention to enforce the consistency of the spliced patch with the surrounding of the removed part and the rest of the image. The overall splicing process is depicted in Fig. 5.

IGPT[19] is a transformer-based unsupervised image classification and image generation model. Early transformer-based models, such as BERT,[44] RoBERTa,[45] and T5,[46] could be used directly with 1D sequences in any form, but were not easily extendable to 2D data, such as

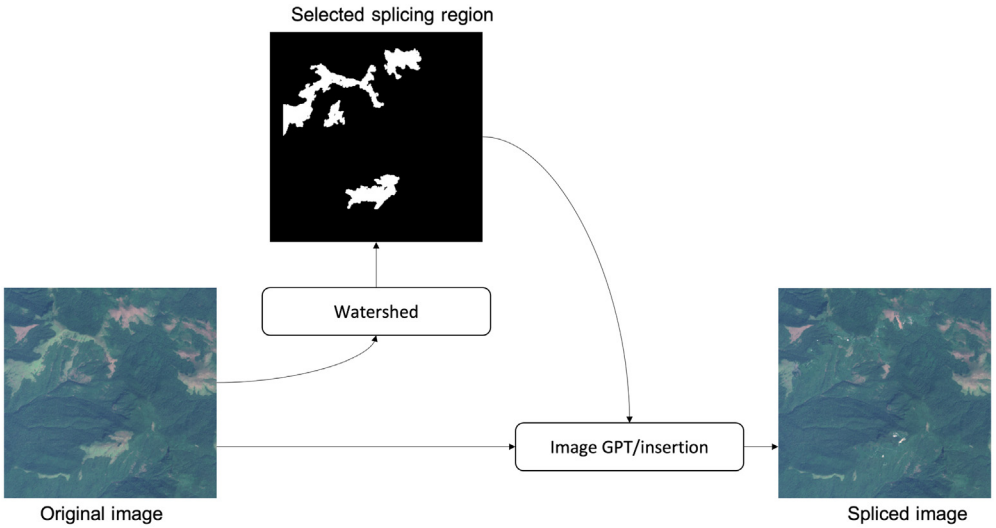

Fig. 5 Spliced images created using iGPT.

images. A model that is able to work with 2D data, namely the GPT-2 [47] model, has been introduced recently (a GPT-2 model trained on images is known as iGPT). To identify the spliced areas, we apply watershed[48] unsupervised segmentation to the images of the world dataset and randomly select several of the largest segments as the regions where the splices generated by iGPT had to be inserted. Then we use iGPT to generate the to-be-inserted splices based on the mask given by the watershed segmentation.

4.3.1 *Architecture*

IGPT[19] is very similar to the GPT-2.[47] An important difference between the two architectures is the activation function. Specifically, a quick Gaussian error linear unit (GELU) is used in iGPT, instead of the GELU used in GPT-2. GELU combines some useful features of the most commonly used activation functions. It randomly multiplies its input by one and randomly sets some of the activations to zero. It can be approximated by the following equation:

$$\mathrm{GELU}(x) \sim 0.5x(1 + \tanh(\sqrt{2/\pi}(x + 0.044715x^3))). \tag{9}$$

Quick GELU is similar to GELU but with a lower computationally complexity, being defined as

$$\mathrm{quick\_GELU}(x) = \frac{x}{1 + e^{-1.702x}}. \tag{10}$$

IGPT also differs from GPT 2 in the number of normalization layers, which is much lower for iGPT, thus decreasing significantly the number of operations. The reader may refer to Ref. 19 for more details. The input of GPT-2 is a sequence of pixels $U = x_1, \ldots, x_n$. Although its objective is to maximize the following likelihood:

$$L_1(U) = \sum_i \log P(x_i | x_{i-k}, \ldots, x_{i-1}, \Theta), \tag{11}$$

where P is the conditional probability modeled by a neural network with parameters Θ, and k is the size of the context window.

The transformer inside the iGPT creates a model of the probability density function of the current pixel x_i, given the previously observed pixels $x_1, \ldots, x_{i-1}$ as shown in the following equation:

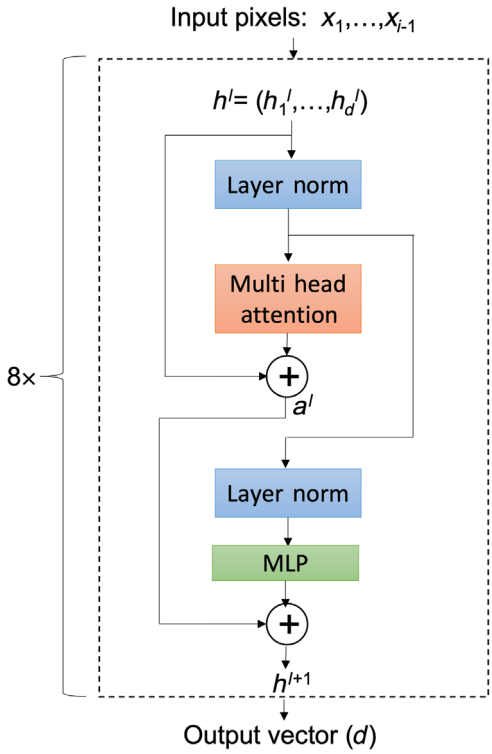

Fig. 6 Block diagram of iGPT.

$$p(x) = \prod_{i=1}^{n} p(x_{\pi_i} | x_{\pi_{i-k}}, \ldots, x_{\pi_{i-1}}; \theta), \tag{12}$$

where in general, π_i indicates a permutation of the pixel sequence. In our case, we simply let π_i be the identity permutation, that is $\pi_i = i$. [19] θ contains all the other parameters of the neural network used during training. The 2D image is transformed into a 1D sequence by lexicographical ordering.

The input of the decoder part of the transformer is a sequence of discrete pixels $x_1, \ldots, x_{i-1}$, and the output is a d-dimensional vector as shown in Fig. 6. The decoder is implemented by a stack of L blocks, in which the l'th block produces an intermediate embedded vector $h_1^l, \ldots, h_d^l$. IGPT uses the same formulation as GPT-2 for the transformer decoder block, in which we input h^l in the order seen in Eqs. (13)–(15) to obtain h^{l+1}:

$$n^l = \text{layer\_norm}(h^l), \tag{13}$$

$$a^l = h^l + \text{multihead\_attention}(n^l), \tag{14}$$

$$h^{l+1} = a^l + \text{mlp}(\text{layer\_norm}(n^l)). \tag{15}$$

The final layer of the transformer decoder is followed by a normalization layer and projection logits (real numbers that with unlimited range) as a parameter for the conditional probability distributions of each sequence element. In the final step, the output vector is reshaped into a 2D image. In our implementation, we used eight layers to construct the iGPT with two heads, we also set the embedded vector dimension to 16, as suggested in the original iGPT paper.[19]

4.3.2 *Training*

IGPT model was trained for 20,000 epochs using the SEN12MS dataset. We had to train the model for a large number of epochs to generate realistic spliced objects. The validation loss was decreasing throughout the entire training process. The size of the patches extracted from the

SEN12MS dataset was $28 \times 28 \times 3$. At each iteration, we extracted one random patch for each image. The model was trained using the negative log likelihood loss function:

$$L(x, y) = \sum_{n=1}^{N} \frac{-\log(e^{x_n - y_n})}{N}, \tag{16}$$

where x_n is the predicted pixel value and y_n is the target pixel value. The target pixel values are the pixel values in the SEN12MS dataset.

For training, we used the Adam optimizer with $\beta_1 = 0.5, \beta_2 = 0.999$ and learning rate equal to 0.003. β_1 and β_2 are the initial decay rates used to calculate the first and second moments of the gradient. The batch size was set to 64. IGPT was trained on the SEN12MS dataset described in Sec. 3.4. Training went on for 4 months on one GPU. 6747 spliced images were generated in ~3 weeks.

4.4 *Splicing with Vision Transformer*

The vision transformer[17] was originally developed for image classification by replacing the convolution layers with a transformer model.[49] Specifically, transformers tend to have inductive bias when trained on large-scale datasets. The vision transformer aims at resolving this issue and provides good results for image classification by scaling the dataset to a smaller size and reducing the amount of training data. In our work, we modified the vision transformer so to use it for synthetic splice generation. In particular, we edited the last layers of the vision transformer, so to generate an image instead of a classification score. The modified layer has a size of $3 \times 256 \times 256$.

We use the modified vision transformer to generate synthetic image splices to be inserted into images from the world dataset. The modified vision transformer was trained on the SEN12MS dataset. The goal here is to remove some regions of an image from the world dataset and replace them with content generated by the modified vision transformer. The content should be consistent with the surrounding of the removed part and the rest of the image. Similar to iGPT, we use watershed[48] segmentation on images in the world dataset and randomly selected several of the largest segments to insert the splices. The whole procedure is shown in Fig. 7.

4.4.1 *Architecture*

The vision transformer inputs are image patches as shown in Fig. 8. In our technique, the original image size is 128×128, whereas the size of the patches inside the transformer is 64×64. Then the dimensionality of the input image patches is reduced using a linear projection: $T_i = W\hat{I}_i$, where $W \in \mathbb{R}^{D \times N}$ is a linear mapping function learned during training, $\hat{I}_i \in \mathbb{R}^N$ is the flattened

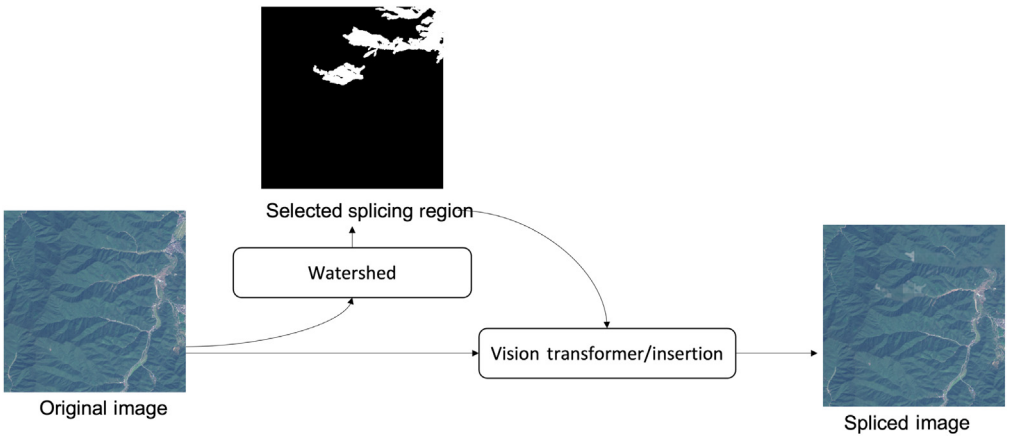

Fig. 7 Spliced images created using vision transformer.

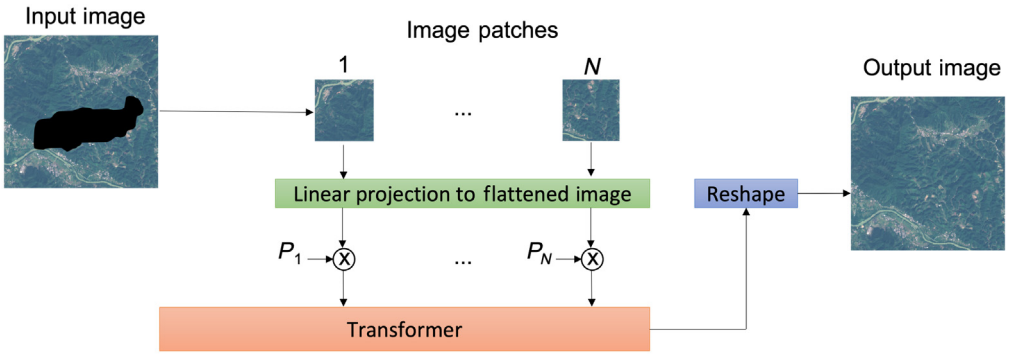

Fig. 8 Block diagram of the vision transformer.

i'th image patch, and $T_i \in \mathbb{R}^D$ is known as the i'th image token. Therefore, an image token is a linear projection of the input patches.

The vision transformer incorporates a traditional transformer used on one-dimensional sequences. The transformers use self-attention modules to incorporate long range information, which can contain information about all the inputs,[49] let us call these inputs "position-aware tokens." The position-aware-tokens indicate where the image patch is located in the input image. The self-attention modules are invariant to the order of the position-aware tokens, so the order with which we input the list of "position-aware tokens" into the transformer is irrelevant. We create the position-aware tokens by concatenating the image token (input image patch projection) and the positional embeddings. The positional embeddings incorporate positional information for each different input image token.[17] We created these positional embeddings by numbering the order of the patches with 64×64 size. These positional embeddings, $P_i \in \mathbb{R}^D$ for $i \in \{0,1,\dots\}$, are used to add positional information about the input patches to the transformer. We input these position-aware tokens into the transformer to produce a "transformer output." We created the output image by reshaping the "transformer output" to the shape of the original image.

4.4.2 *Training*

We trained the vision transformer with images from the SEN12MS dataset, segmented using the watershed algorithm as shown in Figs. 7 and 8. The training loss is defined as the difference between the reconstructed and the original SEN12MS images. The vision transformer was trained for 4000 epochs on $128 \times 128 \times 3$ patches extracted from the 124,511 images of the SEN12MS dataset. For training, we used the Adam optimizer with $\beta_1 = 0.5, \beta_2 = 0.999$, and learning rate equal to 3×10^{-6}. The batch size was set to 1. The vision transformer incorporated a Linformer.[50] The Linformer reduces the memory used in the transformer self-attention module, by reducing the space complexity. The Linformer we used has an internal dimension of 2048, its sequence length is 65 with depth 12. The Linformer uses 1024 heads and introduces a low-rank matrix to approximate the self-attention part in the transformer. In this way, the space and time complexity of the model is reduced to $O(n)$. More detailed information about this architecture can be found in the original Linformer publication.[50] We used the trained vision transformer to generate 285,768 spliced world images. The model was trained for 2 months on one GPU, and the spliced images were generated in ~2 weeks.

5 Results

In the following, we show some examples of global and local image manipulations generated with the models described in the previous sections. For global manipulations, we used γ-correction with γ set to 2 on the RGB bands of the images to visualize them properly since otherwise they would be too dark for human inspection. In addition to γ correction, for the season transfer task, we applied image stretching on the R, G, and B bands of each image. The original images displayed in this section (without corrections) can be accessed in Ref. 51.

The quality of the season transferred images can be judged by comparing them with the available ground truth. (Although comparing the transformed images with the ground truth is a reasonable way to judge the quality of the transfer, it is worth reminding that no unique ground truth exists for the season transfer, given that the same region may assume different aspects in different days of the same season, or across different years.) For the land cover transformation, we judge the quality of the produced images by means of an objective spectral measure such as the normalized difference vegetation index (NDVI)[52] and classifying the image pixels by means of a general purpose classifier.

5.1 *Season Transfer*

In Fig. 9, we show an example of season transfer for the Alps dataset. Real winter and real summer images are shown in columns (a) and (c), respectively, whereas the synthetic generated images are displayed in columns (b) and (d). We report the color image obtained by putting together the RGB bands (first row) and the single-image bands (rows 2 to 5). As it can be seen, the synthetic images produced by the pix2pix model approximate very well the real images, and in any case, they provide a realistic view of the framed region in the target season. A similar example for the Scandinavian dataset is shown in Fig. 10. Even in this case, the synthetic images are very close to the real ones.

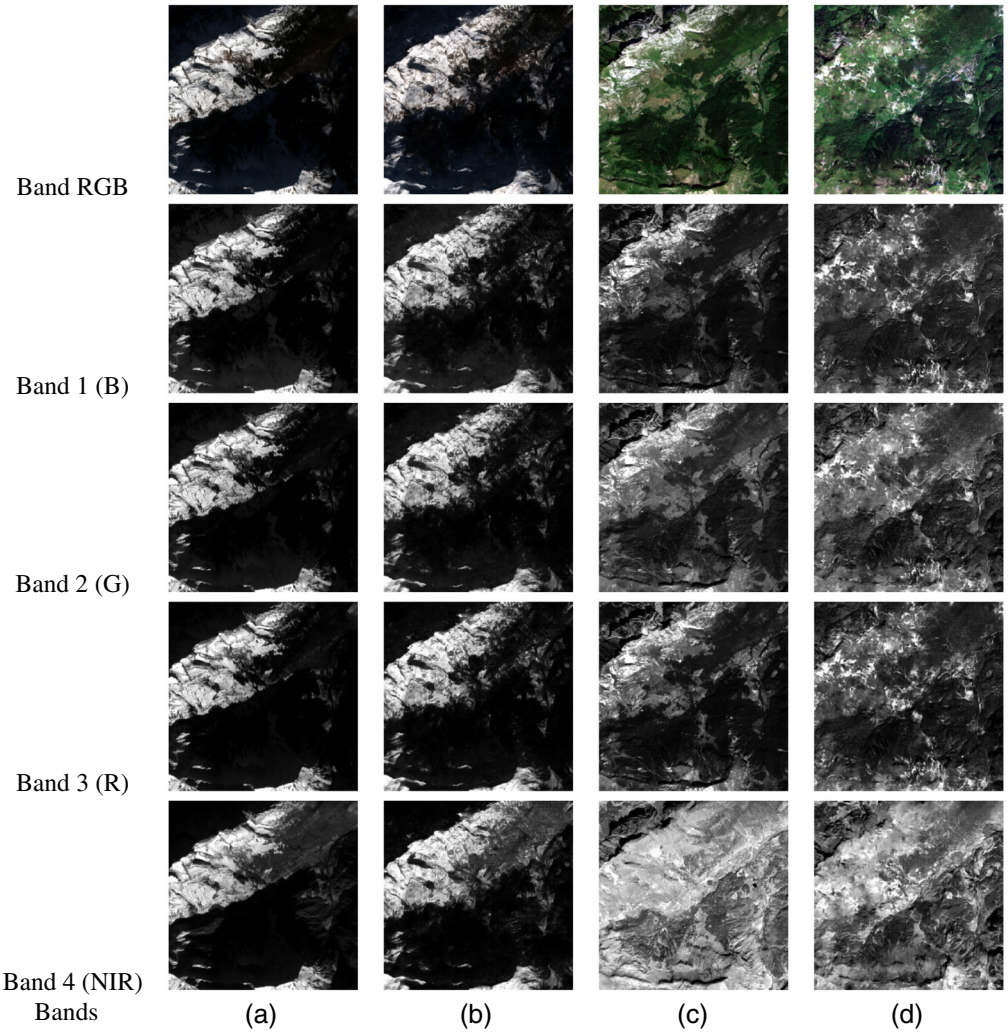

Fig. 9 Alps season transfer example: (a) real winter, (b) generated winter, (c) real summer, and (d) generated summer.

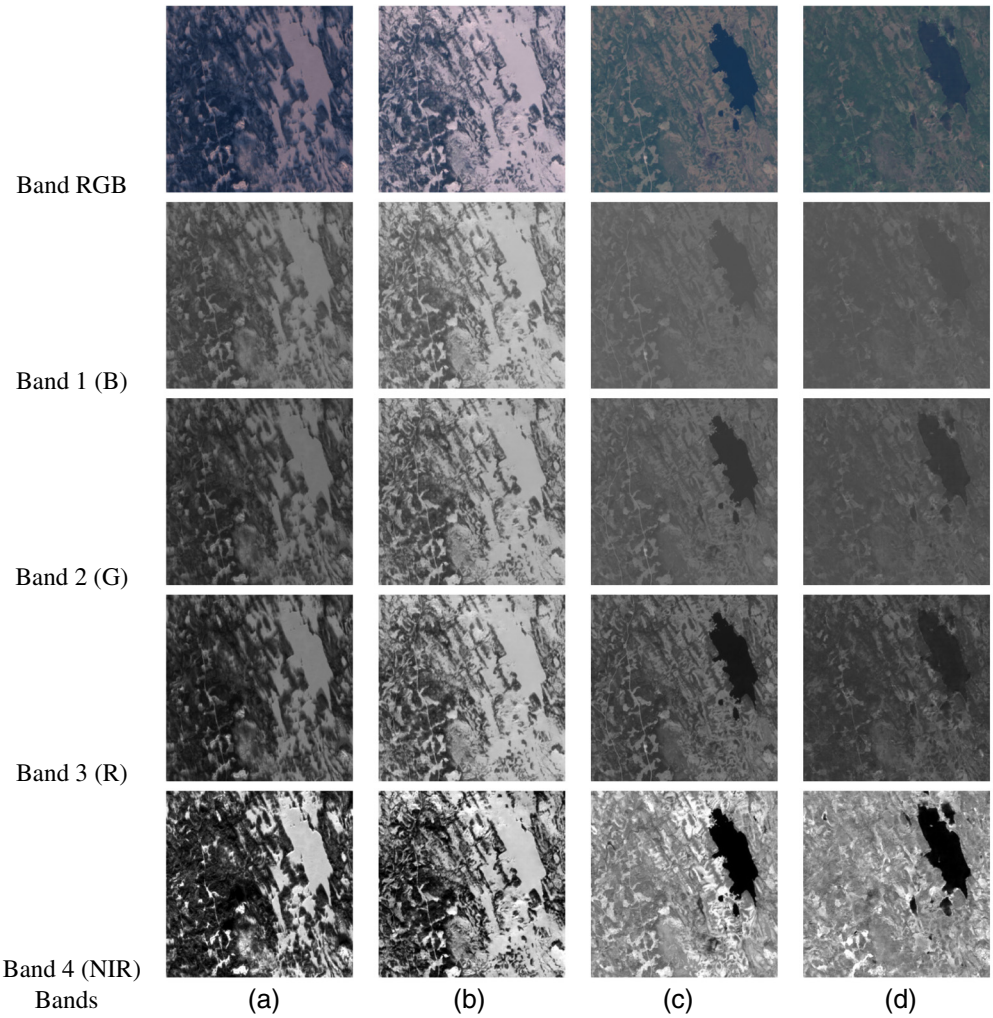

Fig. 10 Scandinavian season transfer example: (a) real winter, (b) generated winter, (c) real summer, and (d) generated summer.

5.2 *Land Cover Transfer*

In Fig. 11, we show an example of a land cover transfer from barren to vegetation and vice versa. We notice that the generated vegetation bands are darker than the input barren images and consistent with the real vegetation. In the same way, the generated barren image is lighter than the vegetation input as it should be for a real barren terrain. To give an objective measure of the effectiveness of the land cover transfer (in the absence of ground truth images), we utilized the NDVI index that is usually adopted to estimate the vegetation content of satellite multispectral images and defined as

$$\mathrm{NDVI} = \frac{\mathrm{nir} - \mathrm{red}}{\mathrm{nir} + \mathrm{red}}, \tag{17}$$

where nir is the near-infrared band (band 4 in our case) and red is the RED band (band 3). We then used the NDVI index to classify each pixel into one of four classes.[53] Specifically, pixels for which the NDVI is lower than -0.1 are classified as water pixels, as barren when NDVI $\in [-0.1, 0.1]$, low vegetation when NDVI $\in [0.1, 0.4]$, and high vegetation when NDVI is larger than 0.4. In Table 2, we report the result of the pixel classification into the above 4 classes for 2000 real vegetation images, 2000 real barren images, 2000 synthetic barren images (GAN barren), and 2000 synthetic vegetation images (GAN vegetation). We confirm that

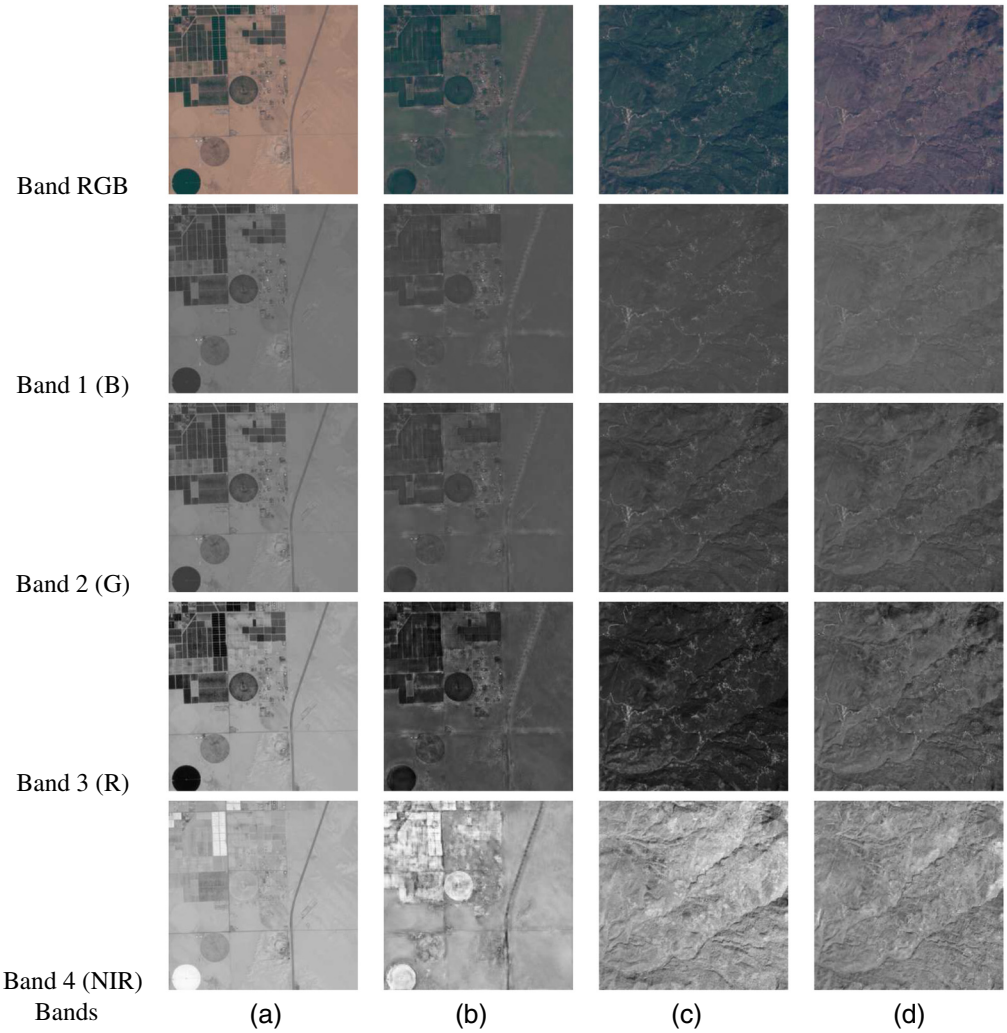

Fig. 11 Land cover transfer examples: (a) real barren, (b) generated vegetation, (c) real vegetation, and (d) generated barren.

Table 2 Percentage of pixels classified into four terrain classes based on NDVI.

	Class			
Dataset	High vegetation (%)	Low vegetation (%)	Barren (%)	Water (%)
Real vegetation	77.87	21.81	0.29	0.01
GAN vegetation	91.8	6.7	1.1	0.3
Vegetation obtained with the model in Ref. 15	68.2	22.3	8.3	1.1
Real barren	0	0	100	0
GAN barren	0	2	97.54	0.46
GAN barren obtained with the model in Ref. 15	11.26	7.6	81.14	0

the majority of the pixels of real vegetation images to the high vegetation class, whereas the majority of the pixels of real barren images to the barren class. For the synthetic vegetation images, most pixels are classified as high vegetation, even if no pixels of the input images belong to such a class. As for the synthetic barren images, most pixels were classified as barren and a few as low vegetation, even if the most input pixels belonged to the high vegetation class. In addition, we compared the results we got with those obtained by applying the NICE GAN model

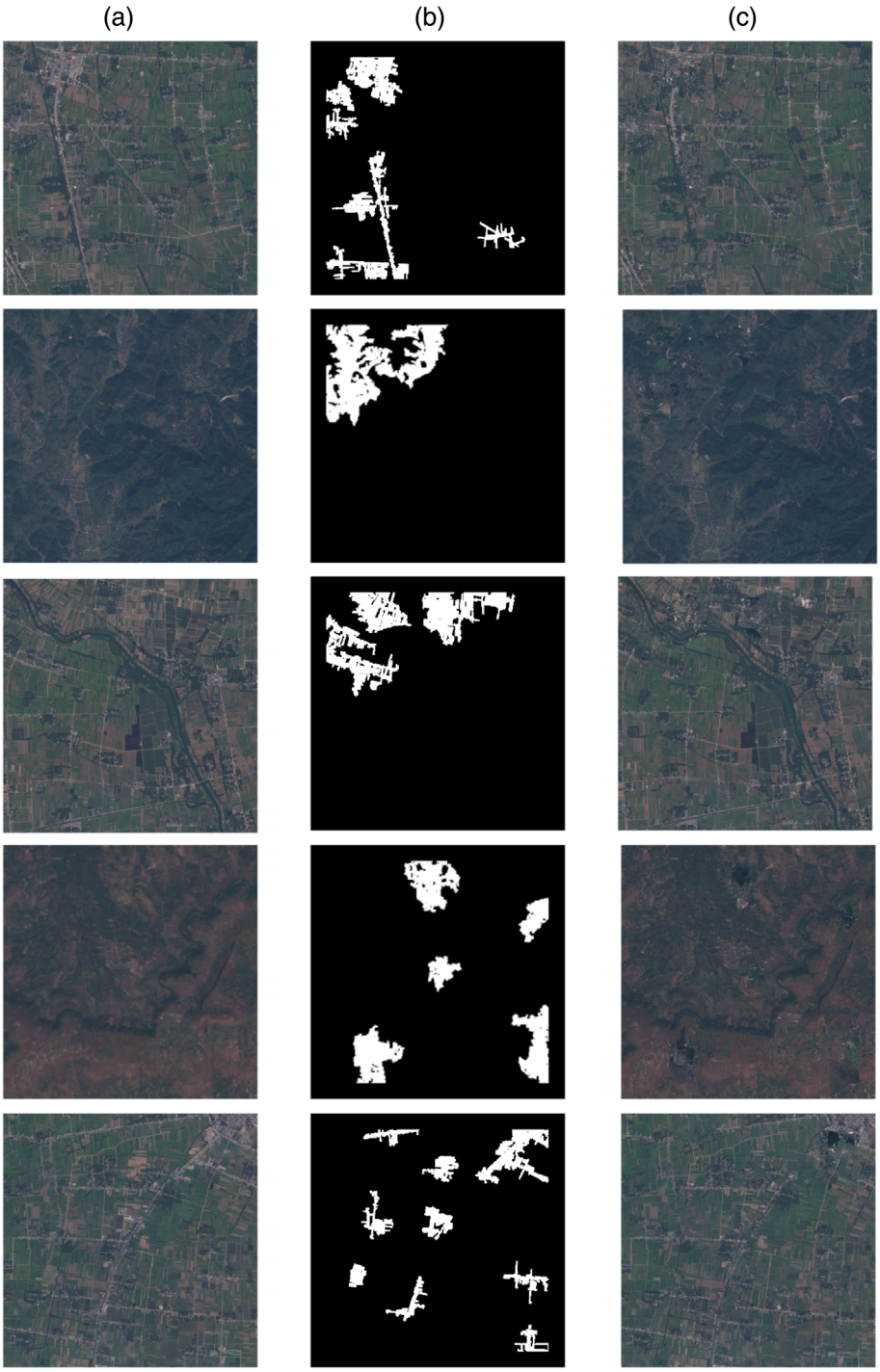

Fig. 12 Example images generated by the iGPT architecture: (a) the original world image, (b) the spliced regions, and (c) the manipulated images.

presented in Ref. 15. To do so, we applied the NICE GAN model to the same pristine images of each class and then we computed the percentage of pixels classified into the four terrain classes similarly to what we did for the GAN images generated by our model. The results show that in the case of GAN vegetation, our model achieves a stronger transfer with only 1.1% of the pixels classified as barren, whereas with[15] 8.3% of the pixels remained in the barren class. As for the GAN barren, our model also shows a stronger transfer capability with more than 97% of the pixels classified as barren, whereas by applying the model described in Ref. 15, a larger number of pixels are still classified as vegetation.

5.3 *IGPT*

In Fig. 12, we show some images containing splices generated by the iGPT model. Figure 12(a) shows the original images, (b) the replacement masks (spliced region), and (c) contains the manipulated images. We can note that the model can generate splices, which blend well into the images. The spliced images contain areas with different climates, vegetations, and urbanization levels. For example, the top urban region in Fig. 13(a) has been synthetically generated, whereas the bottom green vegetation part belongs to the original image. In the same figure, the bottom vegetation part in Fig. 13(b) corresponds to a synthetic region, whereas the surrounding barren pixels are part of the original image.

5.4 *Vision Transformer*

In Fig. 14, we show some examples of the spliced images generated by the vision transformer. Figure 14(a) contains the original images, (b) contains the replacement masks (spliced region), and (c) contains the manipulated images. We generated the spliced regions as in Sec. 5.3. As it can be seen, the vision transformer generates realistic spliced regions. The spliced images contain areas with a different climate including desert, Mediterranean, continental, tundra, and different levels of vegetation. In addition, the spliced regions blend well into the surrounding areas. For example, the bottom forest area in Fig. 15(a) has been synthetically generated, whereas all the other green vegetation parts belong to the original image. In the same figure, the top desert part in Fig. 15(b) corresponds to a synthetic region, whereas the other surrounding desert pixels are original.

We noticed several differences in the splices generated by iGPT and vision transformer. iGPT tends to generated more diverse objects. For example, in Fig. 13(a), the original image was mainly vegetation and rural areas while the generated spliced region is an urban area. IGPT was able to generate splices with a pixel level detail for varying spliced regions, however,

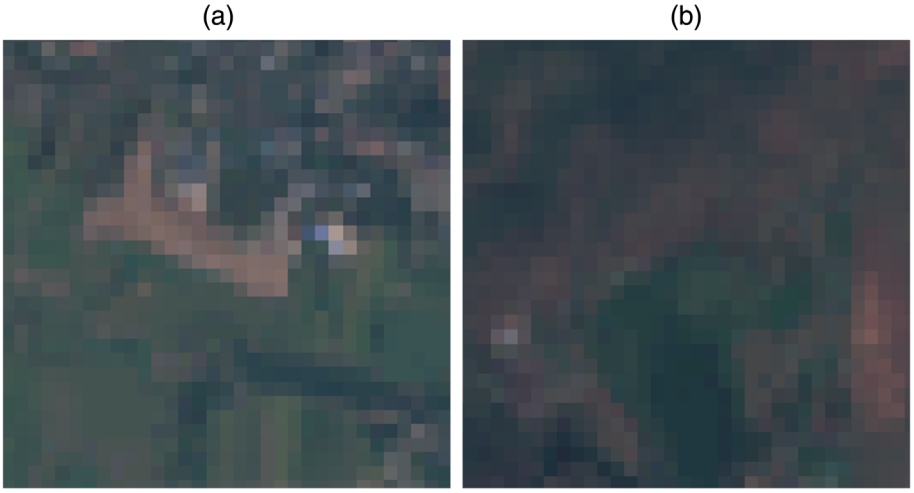

Fig. 13 (a), (b) Generated image parts using iGPT.

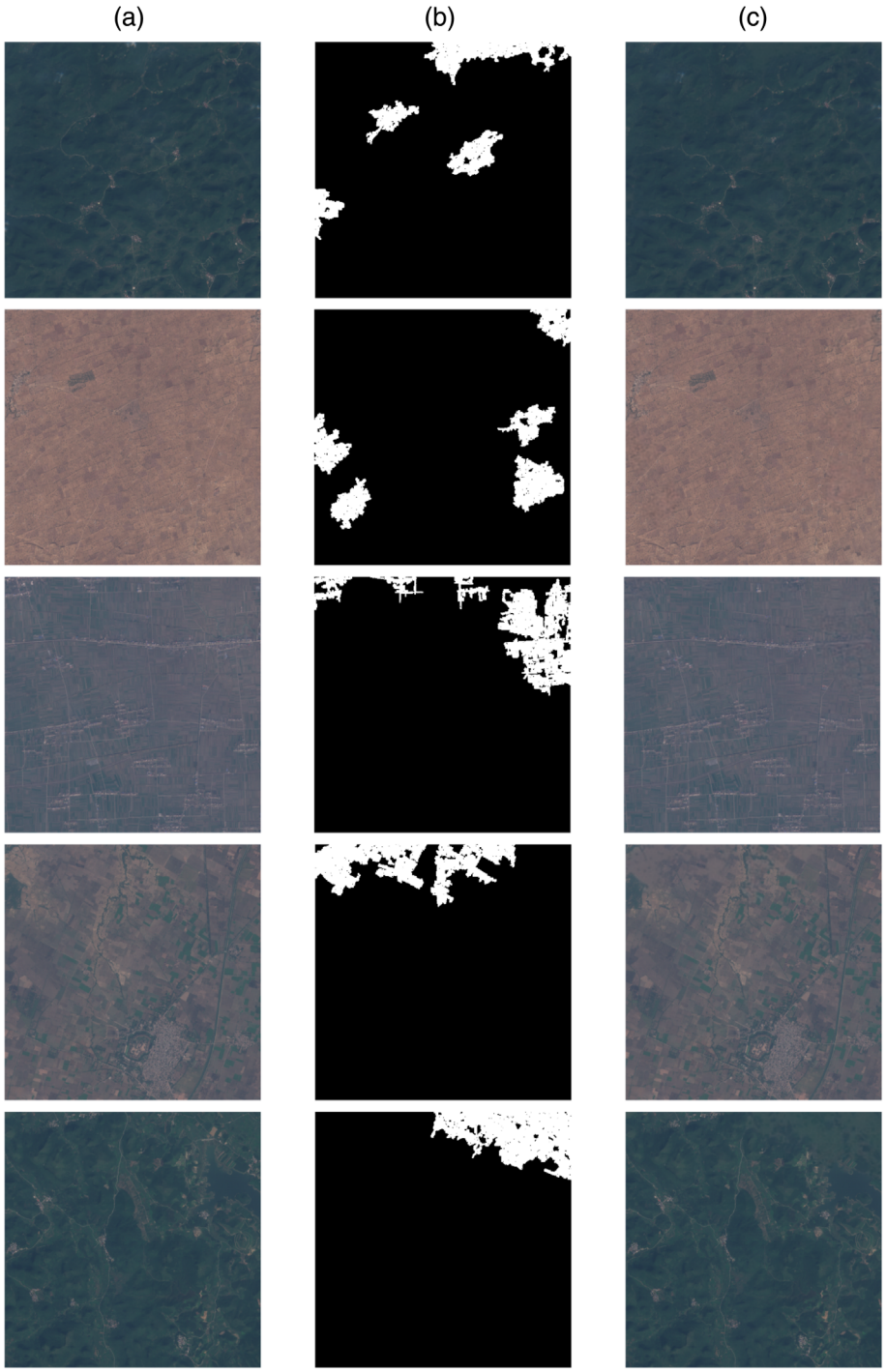

Fig. 14 Examples of images generated by the vision transformer architecture: (a) the original world image, (b) the spliced regions, and (c) the manipulated images.

it took a long time (5 to 30 min) to generate these splices. The vision transformer tends to generate splices very similar to their surroundings. For example, in Fig. 15(a), the original image was a rural housing area, whereas the generated splices were vegetation, which is consistent with the rest of the image that is mainly occupied by vegetation. The vision transformer sometime had some difficulties to generate very detailed spliced regions, however, creating a spliced image took several seconds in contrast to the iGPT long generation time.

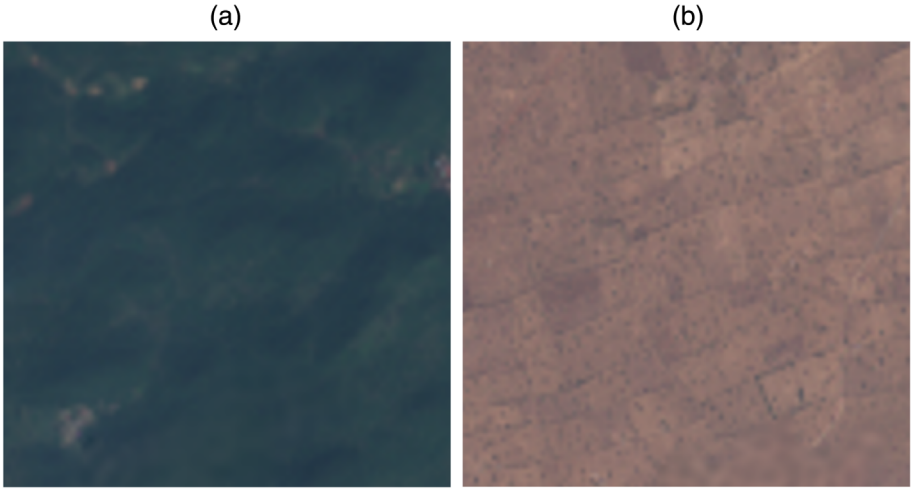

Fig. 15 (a), (b) Generated image regions using the vision transformer.

6 Conclusion

DL generative techniques are able to generate realistic images that can even deceive human inspection. Although, so far, most attention has been given to the generation of face images, we expect that the generation of synthetic satellite images will gain more and more interest in the future. However, conventional techniques used to generate face images cannot be directly applied to create synthetic satellite images, due to the particular nature of satellite multispectral imagery. In this paper, we presented a number of DL architectures aiming at generating labeled synthetically manipulated satellite images. We focused on two kinds of manipulations: full image modification and local splicing. With regard to full image modification, we adapted two GANs commonly used for style transfer applications, to implement two different kinds of transfer: (i) land cover and (ii) season transfer. As to local manipulations, we presented two architectures for local splicing. All the proposed methods can generate highly realistic images, opening the way for several uses across different application scenarios. As future work, we plan to examine whether the synthetic images generated by our architectures can be distinguished from real images by means of specific image forensic detectors.

Acknowledgments

This material is based on research sponsored by the Defense Advanced Research Projects Agency and the Air Force Research Laboratory (Agreement Nos. FA8750-16-2-0173 and FA8750-20-2-1004). The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA and AFRL or the U.S. Government.

References

1. C. Bermudez et al., "Learning implicit brain MRI manifolds with deep learning," *Proc. SPIE* **10574**, 105741L (2018).
2. A. Madani et al., "Chest x-ray generation and data augmentation for cardiovascular abnormality classification," *Proc. SPIE* **10574**, 105741M (2018).
3. M. Fuentes Reyes et al., "SAR-to-optical image translation based on conditional generative adversarial networks-optimization, opportunities and limits," *Remote Sens.* **11**(17), 2067 (2019).
4. W. He and N. Yokoya, "Multi-temporal Sentinel-1 and -2 data fusion for optical image simulation," *ISPRS Int. J. Geo Inf.* **7**(10), 389 (2018).

5. H. J. A. Andrade and B. J. T. Fernandes, "Synthesis of satellite-like urban images from historical maps using conditional GAN," *IEEE Geosci. Remote Sens. Lett.* **19**, 1–4 (2022).
6. A. T. S. Ho and W. M. Woon, "A semi-fragile pinned sine transform watermarking system for content authentication of satellite images," in *Proc. IEEE Int. Geosci. and Remote Sens. Symp.*, Seoul, Vol. 2, pp. 1–4 (2005).
7. S. K. Yarlagadda et al., "Satellite image forgery detection and localization using GAN and one-class classifier," in *Media Watermarking, Secur. and Forensics 2018*, Burlingame, CA, USA, 28 January 2018–1 February 2018, A. M. Alattar and N. D. Memon and G. Sharma, Eds., Ingenta (2018).
8. J. Horváth et al., "Anomaly-based manipulation detection in satellite images," in *IEEE Conf. Comput. Vision and Pattern Recognit. Workshops, CVPR Workshops 2019*, Long Beach, CA, USA, June 16–20, 2019, Computer Vision Foundation/IEEE, pp. 62–71 (2019).
9. E. R. Bartusiak et al., "Splicing detection and localization in satellite imagery using conditional GANs," in *IEEE Conf. Multimedia Inf. Process. and Retrieval (MIPR)*, pp. 91–96 (2019).
10. J. Horváth et al., "Manipulation detection in satellite images using deep belief networks," in *IEEE/CVF Conf. Comput. Vision and Pattern Recognit. Workshops (CVPRW)*, pp. 2832–2840 (2020).
11. D. M. Montserrat et al., "Generative autoregressive ensembles for satellite imagery manipulation detection," in *IEEE Int. Workshop Inf. Forensics and Secur. (WIFS)*, pp. 1–6 (2020).
12. S. Program, "Sentinel missions," https://sentinel.esa.int/web/sentinel/missions
13. I. J. Goodfellow et al., "Generative adversarial nets," in *Proc. 27th Int. Conf. Neural Inf. Process. Syst. (Volume 2, NIPS'14)*, MIT Press, Cambridge, Massachusetts, pp. 2672–2680 (2014).
14. P. Isola et al., "Image-to-image translation with conditional adversarial networks," in *Proc. IEEE Conf. Comput. Vision and Pattern Recognit. (CVPR)* (2017).
15. L. Abady et al., "GAN generation of synthetic multispectral satellite images," *Proc. SPIE* **11533**, 115330L (2020).
16. J. Zhu et al., "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *IEEE Int. Conf. Comput. Vision, ICCV 2017*, Venice, Italy, October 22–29, 2017, IEEE Computer Society, pp. 2242–2251 (2017).
17. A. Dosovitskiy et al., "An image is worth 16×16 words: transformers for image recognition at scale," in *9th Int. Conf. Learn. Represent., ICLR 2021, Virtual Event*, Austria, May 3–7, 2021, OpenReview.net (2021).
18. T. Wolf et al., "Transformers: State-of-the-art natural language processing," in *Proc. of the 2020 Conf. on Empirical Methods in Natural Language Processing: System Demonstrations*, Association for Computational Linguistics, pp. 38–45 (2020).
19. M. Chen et al., "Generative pretraining from pixels," in *Proc. 37th Int. Conf. Mach. Learn., ICML 2020, 13–18 July 2020, Virtual Event, Proc. Mach. Learn. Res.*, PMLR, Vol. 119, pp. 1691–1703 (2020).
20. T. Karras et al., "Progressive growing of GANs for improved quality, stability, and variation," in *6th Int. Conf. Learn. Represent., ICLR 2018*, Vancouver, BC, Canada, April 30–May 3, 2018, Conf. Track Proc., OpenReview.net (2018).
21. T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," *IEEE Trans. Pattern Anal. Mach. Intell.* **43**(12), 4217–4228 (2021).
22. T. Karras et al., "Analyzing and improving the image quality of stylegan," in *IEEE/CVF Conf. Comput. Vision and Pattern Recognit., CVPR 2020*, Seattle, WA, USA, June 13–19, 2020, Computer Vision Foundation/IEEE, pp. 8107–8116 (2020).
23. A. Brock, J. Donahue, and K. Simonyan, "Large scale GAN training for high fidelity natural image synthesis," in *7th Int. Conf. Learn. Represent., ICLR 2019*, New Orleans, LA, USA, May 6–9, 2019, OpenReview.net (2019).
24. M. Schmitt et al., "SEN12MS—a curated dataset of georeferenced multi-spectral Sentinel-1/2 imagery for deep learning and data fusion," *ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci.* **IV-2/W7**, 153–160 (2019).
25. European Space Agency (ESA), "Sentinel-2 level-1c," 2019, https://sentinels.copernicus.eu/web/sentinel/user-guides/sentinel-2-msi/product-types/level-1c

26. P. Ebel, M. Schmitt, and X. X. Zhu, "Cloud removal in unpaired Sentinel-2 imagery using cycle-consistent GAN and SAR-optical data fusion," in *IGARSS 2020-2020 IEEE Int. Geosci. and Remote Sens. Symp.*, pp. 2065–2068 (2020).
27. A. Meraner et al., "Cloud removal in Sentinel-2 imagery using a deep residual neural network and SAR-optical data fusion," *ISPRS J. Photogramm. Remote Sens.* **166**, 333–346 (2020).
28. X. Wen et al., "Generative adversarial learning in YUV color space for thin cloud removal on satellite imagery," *Remote Sens.* **13**(6), 1079 (2021).
29. X. Liu, Y. Wang, and Q. Liu, "PSGAN: a generative adversarial network for remote sensing image PAN-sharpening," in *2018 25th IEEE Int. Conf. on Image Process. (ICIP)*, Athens, Greece, October 7–10, 2018, IEEE, pp. 873–877 (2018).
30. K. Jiang et al., "Edge-enhanced GAN for remote sensing image superresolution," *IEEE Trans. Geosci. Remote Sens.* **57**(8), 5799–5812 (2019).
31. Z. Wang et al., "Ultra-dense GAN for satellite imagery super-resolution," *Neurocomputing* **398**, 328–337 (2020).
32. M. Deudon et al., "HighRes-net: recursive fusion for multi-frame super-resolution of satellite imagery," CoRR abs/2002.06460 (2020).
33. C. X. Ren et al., "Deepfaking it: experiments in generative, adversarial multispectral remote sensing," *Proc. SPIE* **11727**, 117270M (2021).
34. B. Zhao et al., "Deep fake geography? when geospatial data encounter artificial intelligence," *Cartogr. Geogr. Inf. Sci.* **48**(4), 338–352 (2021).
35. R. Chen et al., "Reusing discriminators for encoding: towards unsupervised image-to-image translation," in *2020 IEEE/CVF Conf. on Comput. Vision and Pattern Recognit. (CVPR)*, Seattle, WA, USA, June 13–19, 2020, IEEE, pp. 8165–8174 (2020).
36. The European Space Agency (ESA), "Copernicus open access hub," https://scihub.copernicus.eu/dhus/#/home (accessed 2019–2021).
37. Open Source Geospatial Foundation, "GDAL," https://gdal.org/ (accessed Sept. 2020).
38. OECD, "Land cover in countries and regions," https://stats.oecd.o/Index.aspx (accessed Sept. 2020).
39. S. Z. Li and A. Jain, Eds., *LDA (Linear Discriminant Analysis)*, pp. 899–899, Springer US, Boston, Massachusetts (2009).
40. M. Mirza and S. Osindero, "Conditional generative adversarial nets," CoRR abs/1411.1784 (2014).
41. O. Ronneberger, P. Fischer, and T. Brox, "U-net: convolutional networks for biomedical image segmentation," *Lect. Notes Comput. Sci.* **9351**, 234–241 (2015).
42. D. P. Kingma and J. Ba, "Adam: a method for stochastic optimization," in *3rd Int. Conf. Learn. Represent., ICLR 2015, San Diego, CA, USA, May 7–9, 2015, Conf. Track Proc.*, Y. Bengio and Y. LeCun, Eds. (2015).
43. K. He et al., "Deep residual learning for image recognition," in *IEEE Conf. Comput. Vision and Pattern Recognit., CVPR 2016*, Las Vegas, NV, USA, June 27–30, 2016, IEEE Computer Society, pp. 770–778 (2016).
44. J. Devlin et al., "BERT: pre-training of deep bidirectional transformers for language understanding," in *Proc. 2019 Conf. North Am. Chap. of the Assoc. for Comput. Linguist.: Hum. Lang. Technol., NAACL-HLT 2019, Minneapolis, MN, USA, June 2–7, 2019, Volume 1 (Long and Short Papers)*, J. Burstein and C. Doran and T. Solorio, Eds., pp. 4171–4186, Association for Computational Linguistics (2019).
45. Y. Liu et al., "Roberta: a robustly optimized BERT pretraining approach," CoRR abs/1907.11692 (2019).
46. C. Raffel et al., "Exploring the limits of transfer learning with a unified text-to-text transformer," *J. Mach. Learn. Res.* **21**, 140:1–140:67 (2020).
47. A. Radford et al., "Language models are unsupervised multitask learners," Technical Report, OpenAI (2018).
48. F. M. Serge Beucher, *The Morphological Approach to Segmentation: The Watershed Transformation*, Vol. 1, CRC Press, Boca Raton (1993).
49. A. Vaswani et al., "Attention is all you need," in *Adv. Neural Inf. Process. Syst.*, I. Guyon et al., Eds., Vol. 30, Curran Associates, Inc. (2017).

50. S. Wang et al., "Linformer: self-attention with linear complexity," CoRR abs/2006.04768 (2020).
51. L. Abady, "Image generation samples," https://drive.google.com/drive/folders/1TM_T8PxJBL0a1wqX-69y24UQ6d6N2ls9?usp=sharing (2021).
52. P. Hao et al., "Using moderate-resolution temporal NDVI profiles for high-resolution crop mapping in years of absent ground reference data: a case study of Bole and Manas counties in Xinjiang, China," *ISPRS Int. J. Geo-Inf.* **5**(5), 67 (2016).
53. X. Yuan, J. Tian, and P. Reinartz, "Generating artificial near-infrared spectral band from RGB image using conditional generative adversarial network," *ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci.* **V-3-2020**, 279–285 (2020).

Lydia Abady is currently a PhD candidate in the Department of Information Engineering and Mathematics of the University of Siena. She is currently doing research on using GANs to generate and modify satellite images under the supervision of Prof. Mauro Barni. She is a member of Visual Information Privacy and Protection (VIPP) Group.

János Horváth received his PhD from Purdue University under the supervision of Prof. Edward J. Delp. His research interests revolve around machine learning, computer vision, one class classification, anomaly detection, image/video forensics, and remote sensing.

Benedetta Tondi is currently an assistant professor in the Department of Information Engineering and Mathematics of the University of Siena. She has been an assistant for the course of information theory and coding and multimedia security. She is a member of the VIPP Group led by Prof. Mauro Barni.

Edward J. Delp is the Charles William Harrison distinguished professor of electrical and computer engineering, a professor of biomedical engineering, and a professor of psychological sciences (courtesy). His research interests revolve around image and video compression, multimedia systems, image processing, parallel processing, computer vision, medical imaging and communication, and information theory.

Mauro Barni is a professor in the Department of Information Engineering and Mathematics of the University of Siena. His activity focuses on digital image processing and information security, with particular reference to the application of image processing techniques to copyright protection (digital watermarking) and authentication of multimedia (multimedia forensics).