CloudGAN: Detecting and Removing Clouds from RGB-images using Image Inpainting

Sem Kluiver
Leiden University
Leiden, The Netherlands

Jerry Schonenberg
Leiden University
Leiden, The Netherlands

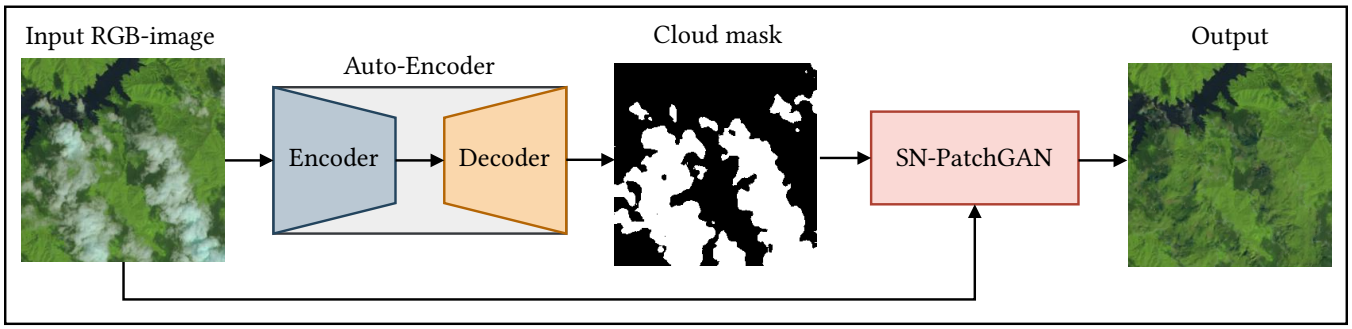


Figure 1. Overview of the CloudGAN, consisting out of a cloud detection and inpainting part. The satellite input image is inserted into the auto-encoder which detects the clouds and generates a cloud mask. This mask indicates the occurrence of clouds pixel by pixel. Next, the cloud mask, along with the input image, is inserted into the SN-PatchGAN [12]. This Generative Adversarial Network is repurposed on inpainting the clouds and gives as output the satellite image with the clouds removed.

Abstract

Remote sensing is an essential tool in monitoring the Earth's surface. However, clouds may obscure the images taken by satellites, thus decreasing the informativeness of these images. Various methods are developed to remove these clouds. They rely on various bands in order to generate a cloud-less image, which may not be available at all times. We present CloudGAN, a method to detect and remove clouds from a satellite RGB-image. Our proposed method uses an auto-encoder for cloud detection, and the SN-PatchGAN image inpainting model for cloud removal. It produces believable results, confirmed by the SSIM and PSNR metrics. Moreover, we demonstrate that the cloud-area threshold, under which the CloudGAN is able to generate realistic cloud-less images, is ~40% for this method. With a greater area covered by clouds, it deviates too much from the ground truth.

Keywords: Cloud Removal, Cloud Detection, Auto-Encoder, Generative Adversarial Network, Image Inpainting

1 Introduction

Remote sensing is a cornerstone of monitoring the Earth's surface. It is used to map and monitor changes occurring throughout the years. However, it is estimated that clouds cover on average 67% of Earth's surface [2]. The clouds obscure images taken from Earth. Under most circumstances, we could wait until a clear shot of the environment can be taken, but in scenarios where that is not possible, we can try to remove the clouds. However, current methods rely on more data than plain RGB-images, such as (non-)visible bands (e.g., infrared), or multiple images taken at different timestamps. This additional data may not be available under all circumstances.

We propose the CloudGAN (Figure 1), a method that is able to inpaint the clouds by using only one RGB-image. In order to achieve this, we have split the task up into two components; cloud detection and inpainting. The first component is handled by an auto-encoder (AE) which generates a cloud mask for a given input image. The second component is done by an image inpainting model, namely the SN-PatchGAN [12]. This model is repurposed for this specific task by training it on a cloud-removal dataset. In short, the paper proposes the following contributions:

1. We propose a simple auto-encoder architecture which can detect clouds from RGB-images.
2. Repurposing the SN-PatchGAN on the cloud-removal problem, and combining it with the AE to form the CloudGAN.
3. Investigating what the threshold is for the area covered by clouds such that cloud-removal is not viable anymore. This threshold is based on the Structural Similarity Index (SSIM) [11], which measures how similar two images are (e.g., the generated image and ground truth).

The remainder of the paper is structured as follows: Section 2 covers the related work; Section 3 gives an overview of the datasets used; Section 4 describes all components of the CloudGAN, while Section 5 discusses the experimental setup.

Section 6 covers the results obtained from the experiments. These results are further discussed in Section 7 and Section 8 contains the overall conclusion of the research.

2 Related Work

Cloud removal is a field with quite some research. Most of the currently available methods, however, rely on more data than only RGB-images. For instance, STGAN [10] requires spatial-temporal data of the same location in order to stitch together an image where no clouds are present, meaning that there needs to be enough images taken until there are no overlapping clouds between all images.

The requirement of data is also present in the cloud detection field. Methods such as s2cloudless [8] require multiple different bands from a satellite. Specifically for s2cloudless, it requires 13 bands from imagery taken by the Sentinel-2 satellite. It determines the cloud probability per pixel based on these 13 bands. Moreover, it does not take neighbouring pixels into consideration when computing the probabilities (in contrary to convolutional neural networks). However, this makes the whole method flexible with regards to input resolutions.

As mentioned in Section 1, the SN-PatchGAN [12] is a component of the CloudGAN. The GAN is retrained on satellite images in order to perform the image inpainting task on images with cloud masks. It is an image inpainting network that features free-form masks and an optional sketch channel, the latter of which allows the user to draw suggestion lines for the network. The SN-PatchGAN also features contextual attention, which enables it to look for textures around the free-form masks. It outperforms other inpainting state-of-the-art methods (e.g., PartialConv [4]), in terms of ℓ_1 and ℓ_2 errors, when evaluated on the original datasets it was trained on (Places2 [13] and CelebA-HQ [5]). Additionally, these results were confirmed by a user study, where the SN-PatchGAN was given higher realness scores than its competitors.

3 Data

This section describes two datasets used to train the AE and SN-PatchGAN with, namely 38-Cloud [6, 7] and RICE [3]. They are intended for the cloud detection and cloud inpainting tasks, respectively.

3.1 38-Cloud

This top down dataset contains 38 high-resolution RGB-images taken by the Landsat 8 satellite along with the ground truth for the cloud detection task. This ground truth describes for each pixel whether it is a cloud or not in the form of a bitmap. These images are divided over a train and test set after which we preprocessed them by applying rotation and cropping transformations. After that, we sampled patches with a resolution of 256×256 from these images and made sure that none of the patches overlapped with each other. This order of creating the dataset is done to minimize the overlap between the train and test set.

After preprocessing, we created 20.023 and 2.352 samples out of the 38 original images for the train and test set, respectively. In total, the dataset contains 22.375 samples. However, the amount of area covered by clouds varies significantly per sample, which is visualized in Figure 2. It shows that the majority of samples have either few or many clouds. Figures 3a and 3b show an example retrieved from the dataset.

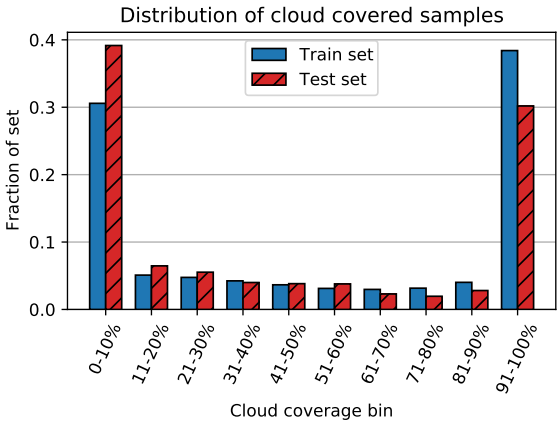


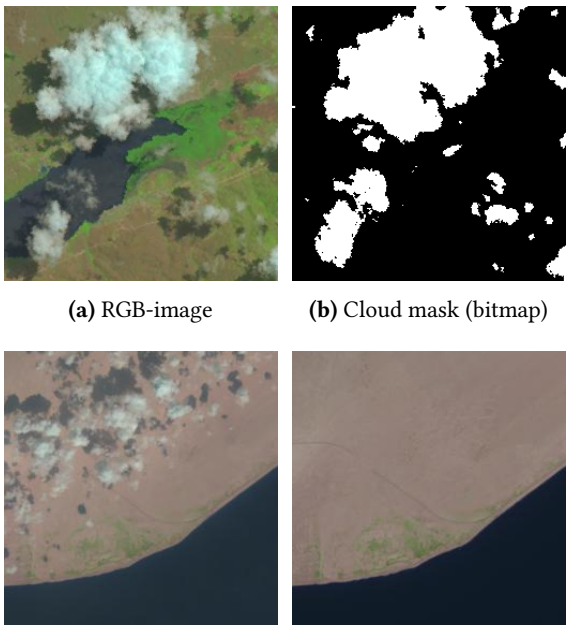
Figure 2. Distribution of the degree of cloud coverage per sample from the train and test sets. It shows that the majority of the samples are in one of the outmost bins.

3.2 RICE

The Remote sensing Image Cloud rEmoving (RICE) dataset is aimed at cloud removal tasks. It provides RGB-images of landscapes with and without clouds. Moreover, the RICE dataset can be divided into the RICE-I and RICE-II subsets. The RICE-I subset is collected on Google Earth and contains 500 pairs of samples. These samples are generated by toggling the cloud-option in Google Earth on and off. This makes the RICE-I subset an intermediate dataset.

As for the RICE-II subset, it is retrieved from the Landsat 8 OLI/TIRS dataset and consists out of 736 pairs. Here, the temporal-difference between every pair of images is at most 15 days. Consequently, minor differences in landscape and/or color can occur between a pair. Examples from this dataset are shown in Figures 3c and 3d. This subset is a top down dataset.

Originally, the samples have a resolution of 512×512, but to be consistent with the 38-Cloud dataset and increase the number of samples, the samples are split up into four equal parts (resulting in a resolution of 256×256). After splitting, we have a total of 4.944 image pairs, of which 3.956 are in the training set and 988 are in the test set.



(a) RGB-image **(b)** Cloud mask (bitmap)

(c) RGB-image with clouds **(d)** RGB-image without clouds

Figure 3. A pair from both the 38-Cloud (**a**, **b**) and RICE datasets (**c**, **d**). Images (**a**) and (**b**) represent an input and cloud mask, respectively. From the RICE dataset, subfigures (**c**) and (**d**) show an image with and without clouds. Note how these images slightly differ in color since they are taken at different points in time (with a maximum temporal difference of 15 days).

4 CloudGAN

Removing clouds from a satellite image consists of two components; cloud detection and inpainting. They are combined into one pipeline which will generate an image without clouds, referred to as the CloudGAN. See Figure 1 for a visualization of the CloudGAN, along with input and output images from the two components.

In the first component, the CloudGAN has to convert the satellite image (Figure 3a) into a cloud hard mask (Figure 3b). In these hard masks, `1` and `0` denote a pixel with and without clouds, respectively.

Next, the cloud mask is subtracted from the satellite image, which is then inserted into an inpainting method. For this, we repurposed the SN-PatchGAN [12] to fill in the gaps. We chose this method since it allows to insert free-form masks, which is essential due to the fact that clouds are not restricted to any shape.

4.1 Cloud detection

The first part of removing clouds from satellite images is to mark the clouds in a bitmap. This is done by an auto-encoder, with its architecture denoted in Table 1.

The generated bitmap denotes, pixel-wise, where the clouds are located in the RGB-image. Pixels with a value of `1` and `0` correspond to pixels with and without clouds, respectively. However, the AE uses sigmoid as its final activation function, meaning that it outputs a soft mask of the input (i.e., the probability of a pixel being a cloud). Consequently, the output has to be converted into a hard mask by rounding all pixels to the nearest integer. Important to note is that the AE is trained on the soft-mask with Mean Squared Error (MSE) as its loss-function. Lastly, the Adam optimizer with a constant learning rate of 0.001 is used to update the weights throughout the training procedure.

For all components of the AE holds that the default value from TensorFlow 1.15 is used as value for all unspecified hyperparameters.

Table 1. The auto-encoder architecture. Every convolutional layer has a kernel size of 3×3 and strides of 2. Moreover, the padding for every layer is set to the value '`same`'. In total, the model has 198.593 trainable weights. The layer names refer to the layers used by TensorFlow.

Layer	Filters	Activation	Output shape
Input	N.A.	N.A.	(256, 256, 3)
Conv2D	128	ReLU	(128, 128, 128)
Conv2D	64	ReLU	(64, 64, 64)
Conv2D	32	ReLU	(32, 32, 32)
Conv2DTranspose	32	ReLU	(64, 64, 32)
Conv2DTranspose	64	ReLU	(128, 128, 64)
Conv2DTranspose	128	ReLU	(256, 256, 128)
Conv2D	1	Sigmoid	(256, 256, 1)

4.2 Cloud inpainting

The second part of removing clouds from satellite images consists of combining the generated cloud mask and input image and inserting it into the SN-PatchGAN. The SN-PatchGAN will then proceed to fill in the missing information of the image, which corresponds to the location of the clouds; the clouds are substituted by the inpainted patches the SN-PatchGAN generates. Due to the support of free-form mask of the SN-PatchGAN, we are able to create cloud masks that can have any shape or form; we are not limited to creating square-shaped masks. This is also the main motivation behind using the SN-PatchGAN. In addition, it features gated convolutional layers, which, in short, are learned soft-masks rather than hard masks used in other inpainting models. The model also utilises contextual attention, which learns what textures from the patches around the mask. These textures are then used to reconstruct the missing part in the image. These last two features resulted in SN-PatchGAN outperforming other image inpainting models (e.g., PartialConv [4]) in terms of ℓ_1 and ℓ_2 errors, making it a state-of-the-art inpainting model. All these factors combined contributed to selecting the SN-PatchGAN for the cloud removal step.

5 Experiments

Each component of the CloudGAN is evaluated by means of experimentation. The AE is trained on the 38-Cloud dataset, while the SN-PatchGAN is trained on the RICE dataset plus all images from 38-Cloud that do **not** contain any clouds. This is because the SN-PatchGAN stochastically generates the masks by itself during training. Moreover, data augmentation is applied to each of the train sets in the form of random rotation (max. 180°), shifting (max. 10%), cropping (max. 20%) and flipping (both horizontally and vertically).

5.1 Metrics

The AE is evaluated based on three metrics: Mean Squared Error (MSE), recall, precision and F1-score. MSE is computed on the soft-mask generated by the AE, while the recall and precision are based on the hard-mask which is obtained by rounding each pixel of the soft-mask. The recall denotes how many of the cloud-pixels were predicted correctly, while the precision indicates how many of the non-cloud-pixels were predicted correctly.

As for the SN-PatchGAN, it is evaluated with the Structural Similarity Index (SSIM) [11] and Peak Signal-to-Noise Ratio (PSNR) metrics. SSIM denotes the perceptual difference between the generated and target images. The value of SSIM lies in the interval $[0, 1]$, where 1 indicates that the two images are identical. Equation 1 formally defines the SSIM metric, where x and y are the predicted and target images, respectively. Moreover, μ_i denotes the mean of image i, σ_i the variance of image i, σ_{xy} the co-variance of x and y. Then, $c_1 = (k_1 L)^2$ and $c_2 = (k_2 L)^2$ which are used for stability (to prevent division by zero). Here, k_1 and k_2 are tuneable settings, which are set to 0.01 and 0.03 respectively. Lastly, L is set to $2^{\#\text{bits per pixel}} - 1$.

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

As for PSNR, it computes the overall difference between two images based on the MSE. Equation 2 denotes the PSNR metric, where MAX_I is the maximum value of a pixel (i.e., 255). It expresses the noise in terms of dB.

$$\text{PSNR}(x, y) = 20 \cdot \log_{10}\left(\frac{\text{MAX}_I}{\sqrt{\text{MSE}(x, y)}}\right) \tag{2}$$

This means that the higher the PSNR-score, the more similar the predicted and target images are. However, if x and y are identical, then PSNR is undefined due to $\text{MSE}(x, y) = 0$.

5.2 Setup

As mentioned earlier, each component of the CloudGAN is evaluated separately. The hard-mask output of the AE is compared against the target masks, while the SN-PatchGAN itself is evaluated by combining a cloud mask and cloud-less image from the RICE dataset. The output from the SN-PatchGAN is then compared against the cloud-less image. This is done to get a fair evaluation for each of the components. For instance, it would be unfair to evaluate the SN-PatchGAN with a mask generated by the AE which is partially incorrect, this would make the performance of the SN-PatchGAN seem worse than it is.

The performance of the SN-PatchGAN is partly dependent on how much information is available for the model, which depends on the size of the area covered by clouds. Therefore, we investigate the performance of the SN-PatchGAN on different cloud coverage bins as shown in Figure 2 separately. All 1.815 cloud-less images from the combined RICE- and 38-Cloud test sets are selected and then combined with 46 non-empty masks from each bin. Then, the difference between the inpainted result and the target image is calculated with the SSIM score in order to evaluate the inpainting performance. The SSIM scores will be shown in a violin plot, which should provide a good impression of how well the SN-PatchGAN performs depending on the cloud coverage. The violin plot will also show the SSIM score between the ground truth and the input image with the mask applied, but without inpainting (so without filling in any of the gaps). We consider this as the lower bound SSIM score for that particular image with that particular mask.

The AE and SN-PatchGAN models are trained on a machine with the following specifications:

- Intel Xeon E5-2650 @ 2.00 GHz (32 threads)
- 64 GB RAM
- NVIDIA GeForce RTX 2080 Ti (11 GB VRAM)

The AE is trained for 64 epochs, while the SN-PatchGAN is trained for 1.1M iterations. The SN-PatchGAN uses the default parameters provided in the original source code [1], except for the batch size, which was reduced to 12 due to GPU memory limitations. Both are trained on TensorFlow 1.15. The code is available at https://github.com/JerrySchonenberg/CloudGAN.

5.3 Baselines

The performance of the AE is compared against various baselines. It is evaluated against a naive approach, pix2pix [1] and U-Net [9]. Pix2pix was chosen due to its flexibility: it supports many different Image-to-Image translation modes. U-net was chosen because of its more advanced network structure and performance on image segmentation tasks.

The naive approach determines whether a pixel is a cloud based on its HSV-values. First, the probability for a pixel to be a cloud based on its HSV-values is computed over the train set. These probabilities are separated over the three components (Hue, Saturation and Value), which are visualised in Figure 4.
Given these probabilities, predictions of new pixels are made

[1] https://github.com/JiahuiYu/generative_inpainting

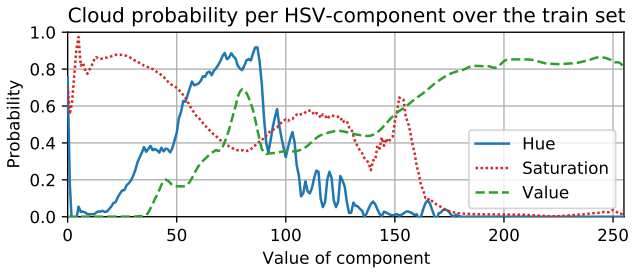


Figure 4. Probability of a pixel representing a cloud per HSV-component. These probabilities are used by the naive approach to determine whether a pixel is a cloud, so the prediction is purely based on the pixel itself, not taking the neighbouring pixels into consideration. If the mean over the three component-probabilities is higher than a threshold τ, then it is predicted to be a cloud-pixel. Important to note is that the Hue lies in the range $[0, 180]$ while the Saturation and Value components lie in the $[0, 255]$ interval.

based on Equation 3, where p is the prediction, h, s, v are the HSV-components, (x, y) are the coordinates in the image, and τ is the threshold. We set the threshold to 0.5. Note that the naive approach does not contain any stochastic elements, with as consequence that we cannot conduct multiple replications for this method.

$$p_{y,x} = \mathbb{1}\{(h_{y,x} + s_{y,x} + v_{y,x}) \,/\, 3 \geq \tau\} \tag{3}$$

Another baseline is the pix2pix[2] method, which is an Image-to-Image Translation method using Conditional Adversarial Nets. Lastly, U-Net is a method originally aimed at image segmentation but is repurposed on the cloud detection task (which is a form of image segmentation). Important to note is that the size of the U-Net architecture is decreased significantly; we divided the number of filters per convolutional layer (barring the final two layers) with 2 twice (e.g., 512 becomes 128). Both baseline methods are trained on the same 38-Cloud dataset as the AE. Pix2pix and U-Net are trained for 100 and 200 epochs, respectively.

We have chosen to not compare the SN-PatchGAN to other methods, as it is a state-of-the-art method. It is performing better than other inpainting models it was compared to, both in terms of ℓ_1 and ℓ_2 errors, as well as in a user study. It therefore did not seem necessary to compare SN-PatchGAN to another inpainting method.

6 Results

This section contains the results from the experimental setup described in the previous section. First, the two components of the CloudGAN, cloud detection and inpainting, are discussed, after which the cloud coverage threshold is determined.

[2]Implementation taken from: https://github.com/affinelayer/pix2pix-tensorflow

6.1 Cloud detection

Firstly, we evaluate the AE on the test set of the 38-Cloud dataset. The model is compared against the following methods: Naive, pix2pix [1] and U-Net [9]. Table 2 denotes the MSE, recall, precision and F1-score for the methods. For the trainable methods, we averaged over three replications. It shows that the U-Net baseline outperforms all other methods in terms of all four metrics. However, the difference in performance between the U-Net and AE is minor and is explained by the significant difference in network capacities; the U-Net contains, despite our efforts to reduce its size, 1.941.381 trainable weights. When comparing it to the 198.593 weights for the AE, they differ around a magnitude of 1 in terms of size.

Table 2. Quantitative results on the test set of 38-Cloud. Every trainable method is replicated three times, with its mean and corresponding 95% confidence interval reported. The MSE is computed over the soft mask.

Method	MSE	Recall	Precision	F1-score
Naive	N.A.	0.877	0.746	0.806
Pix2pix [1]	0.353 ± 0.005	0.568 ± 0.024	0.564 ± 0.038	0.566 ± 0.029
U-Net [9]	**0.090 ± 0.002**	**0.907 ± 0.043**	**0.850 ± 0.017**	**0.877 ± 0.012**
AE (ours)	0.097 ± 0.014	0.888 ± 0.012	0.840 ± 0.013	0.860 ± 0.008

Figure 5 shows qualitative results of the AE on five images taken from the test set. The soft mask prediction is the direct output from the AE and contains a lot of uncertainty. However, when converting it to a hard mask, it resembles the ground truth. In these hard masks, we have visualized the pixels which were incorrectly predicted to be clouds with red, and pixels which were incorrectly predicted to not be clouds with cyan. In other words, the cyan plus white parts from the hard mask equals the ground truth.

Moreover, the results show that the AE struggles with the edges of the clouds (all cyan parts are on the edge of the white parts), this is due to the fact that the edges are less opaque than the center of the clouds and diffuse with the non-cloud area. However, when manually comparing the predictions with the corresponding inputs, it becomes clear that, even for humans, detecting those edges is difficult since it is almost indistinguishable with the non-cloud pixels. Overall, the AE is able to detect the opaque clouds and struggles with the thin clouds. The red parts are less critical for the CloudGAN since it will fill those in when necessary. Only when these areas become too big, it might miss useful information in order to reconstruct the image.

6.2 Cloud inpainting

Secondly, we evaluate the performance of the SN-PatchGAN in inpainting the clouded images. The performance of method is measured by their SSIM and PSNR values of the output images to the ground truth. The average SSIM and PSNR

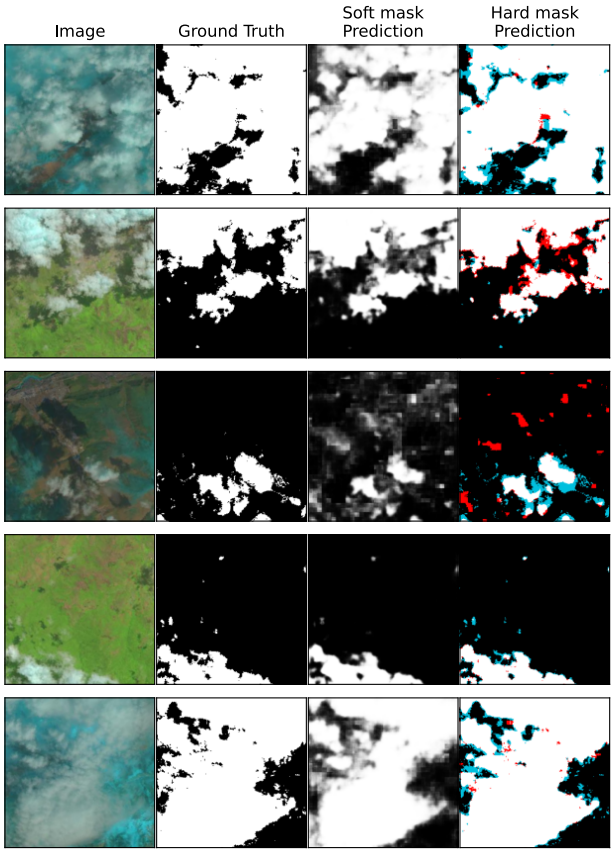


Figure 5. Qualitative results of the AE on samples from the test set. It shows how the generated soft mask is transformed into the hard mask. Additionally, the red areas in the hard mask prediction denote the pixels that were predicted incorrectly to be clouds. Similarly for the cyan areas, here the pixels that are predicted incorrectly to not be clouds are marked.

of these models can be seen in Table 3. The (**1**) denotes the model where the original, clouded image was used as the input image. The (**2**) indicates that the ground truth image was used was the input image, with the cloud mask applied (i.e., simulating perfect cloud detection). In other words, (**1**) is the actual CloudGAN, while (**2**) is solely the SN-PatchGAN.

The table shows that CloudGAN (**2**) significantly outperforms CloudGAN (**1**) over both metrics. This difference is visualized in Figure 6, which shows the qualitative results of four test-images inpainted by the models. For CloudGAN (**1**), we can see that the results between CloudGAN (**1**) and (**2**) vary significantly. This is due to the fact that the AE does not detect the shadows and edges of the clouds. For instance, notice the difference between the images of the fourth row. Missing the small edges of the clouds can deteriorate the inpainting process significantly, due to the contextual attention used by the SN-PatchGAN. The gaps are filled in with textures surrounding it, and in this case these textures belong to the remaining parts of the clouds/shadows. Moreover, the qualitative results also show a limitation of the AE: it is not trained to detect the shadows created by the clouds. These shadow-textures are then reused by the SN-PatchGAN when inpainting the image (rows 2 and 3 visualise this effect). When applying the cloud mask on the ground truth, where there are no cloud edges and shadows present, we can see that the SN-PatchGAN generates images which resemble the ground truth.

Table 3. Mean and standard deviation denoted for the CloudGAN methods. These methods are evaluated on only the test set from RICE, since the 38-Cloud samples do not contain cloud-less images. Note that the number of replications here is 1, due to the extensive training time required for the SN-PatchGAN.

Method	SSIM	PSNR
CloudGAN (**1**)	0.629 ± 0.248	18.630 ± 5.942
CloudGAN (**2**)	0.699 ± 0.284	25.575 ± 14.578

Overall, Figure 6 illustrates the importance of having a cloud mask that covers every cloud. Otherwise, these errors are adopted by the SN-PatchGAN and used in the inpainting process, resulting in the shown artefacts.

6.3 Cloud coverage threshold

The performance of the SN-PatchGAN on different percentages of cloud coverage expressed in SSIM scores can be seen in Figure 7. Note that the violins show the distribution of the SSIM score per bin. The width of the violin is therefore not a representation of how many samples actually achieved a certain SSIM score. The middle black bar in each violin plot shows a small boxplot.

It becomes clear that the SSIM score distribution of the SN-PatchGAN generated images is spread out more the higher the cloud coverage is. The lower bound linearly decreases when the cloud coverage increases. The peak of the distribution also shifts to lower SSIM scores the higher the cloud coverage is. This is not unexpected, since a higher cloud coverage results in more data generated by the SN-PatchGAN, which again results in an image deviating more from the ground truth. We suggest that the quality of images becomes too unreliable with a cloud coverage of 41% or higher. Notice how there is a peak indicating high SSIM values in almost every bin. We suspect, however, that these high SSIM values are achieved on images without any, or very little features, such as images of the sea. One example can be seen in Figure 6: the last row shows an image of the sea with a high cloud coverage, and CloudGAN (**2**) is able to create an image resembling the ground truth.

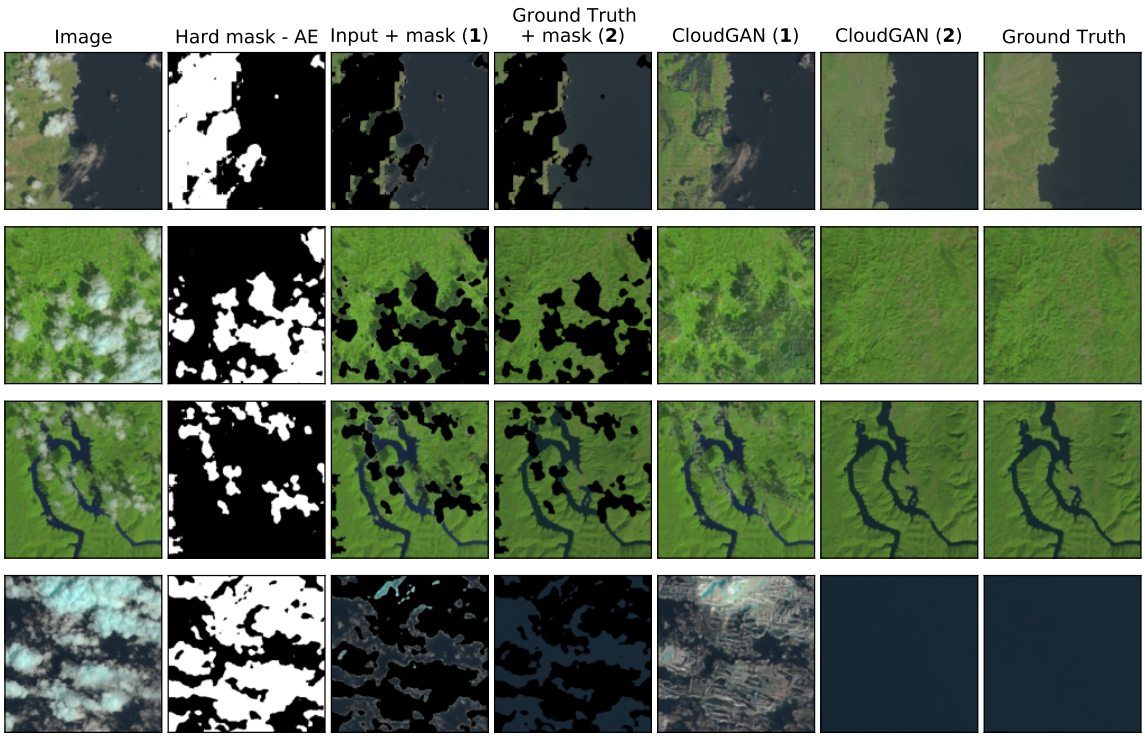


Figure 6. Qualitative results of CloudGAN on four different images. The first and last columns show the clouded input image and ground truth, respectively. The second column shows the mask generated by the AE. The (**1**) and (**2**) denote the CloudGAN being applied on the clouded input image, or on the ground truth, with the mask applied. The third and fourth columns show the input with mask subtracted, while the fifth and sixth show the generated output.

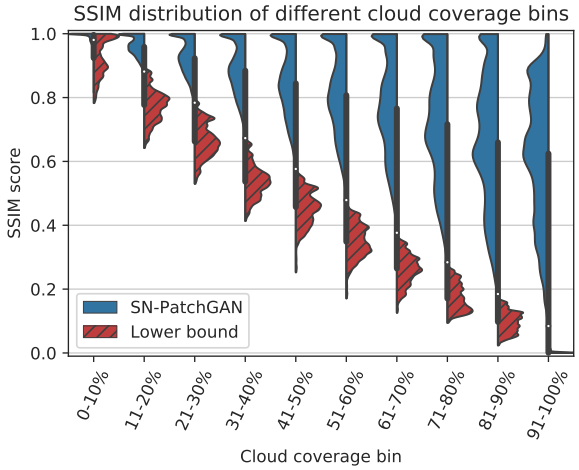


Figure 7. SSIM score distribution over all bins. Important to note is that the density of every pair of violin plots is different, in order to ensure readability. This means that we can only compare pairs of violin plots with each other, and not across bins. However, the plots still show the decrease in quality the more clouds are present in the image. The middle black bar in each violin plot is a boxplot.

7 Discussion

As mentioned in Section 5, the image inpainting part performs quite well under most circumstances. The inpainted satellite images look realistic, albeit slightly different from the ground truth. There are some cases where the cloud removal fails, or performs less well:

- **Foggy, semi-transparent clouds are not recognized by the AE.** This results in some of the generated images looking more faded, because the semi-transparent clouds have not been removed. As described before, masking (all) foggy clouds is not an option because the SN-PatchGAN will not get enough information. Other methods would have to be investigated to address this problem.
- **Incorrect textures are propagated during inpainting.** Considering the AE does not detect all cloud edges and shadows correctly, cloud and shadow textures are propagated and give unrealistic results due to the use of contextual attention.
- **Images with a high cloud coverage result in unrealistic inpainted images.** Due to the high amount of missing information, the SN-PatchGAN has difficulty with inpainting the missing parts of the image

and deviates significantly from the ground truth. The contextual attention module will not perform as well when a large part of the image is missing information, since it has less texture patches to choose from.

- **The inpainting of human-made structures is more difficult for the SN-PatchGAN to get right.** Square and/or systematic, non-organic structures created by humans are more complex than mountains, rivers and hills. This may also be due to the fact that there is a relatively low number of images containing human-made structures in the 38-Cloud and RICE datasets when compared to organic structures.

Moreover, it should be taken into consideration that the temporal difference between some pairs of images in the RICE-II subset results in the SSIM and PSNR metrics being reported worse than they should be. However, this difference should be minimal since not all pairs differ significantly.

In terms of future work, the AE could be improved to recognize cloud shadows and edges to create a more neutral image. This may, however, also result in the loss of more information. The SN-PatchGAN can profit from more training samples, especially of human-made structures. Datasets of images both with and without clouds are, however, sparse. The faded images due to foggy, semi-transparent clouds not detected could be improved by saturation changes. Consequently, training the auto-encoder to recognize these semi-transparent clouds will probably result in masks that are too large for the SN-PatchGAN to create a sensible image with. Meaning that other methods suit these type of images more than CloudGAN.

Additionally, the sketch channel of the SN-PatchGAN could be used to annotate some (semi-transparent) parts of the images, which could help with inpainting the images. The sketch channel could even be automatically generated if the cloud is semi-transparent. It could work similarly to pix2pix' "aerial to map" mode [1].

8 Conclusion

We have presented CloudGAN, a method to remove clouds when only RGB satellite image data is available. The cloud removal method consists of two parts: cloud detection, which is implemented with an auto-encoder, and cloud removal, which uses a state-of-the-art image inpainting model called SN-PatchGAN [12]. By means of experimentation, we show that the quality of the resulting images depends on the area covered by clouds. We suggest that the CloudGAN is unable to provide quality images when more than ~40% of the input image is covered by clouds. In addition, the auto-encoder is sometimes unable to detect cloud shadows and edges, which results in wrong textures being used for inpainting and significantly deteriorating the end result.

References

[1] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. 2017. Image-to-Image Translation with Conditional Adversarial Networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. IEEE Computer Society, 5967–5976. https://doi.org/10.1109/CVPR.2017.632

[2] Michael D. King, Steven Platnick, W. Paul Menzel, Steven A. Ackerman, and Paul A. Hubanks. 2013. Spatial and Temporal Distribution of Clouds Observed by MODIS Onboard the Terra and Aqua Satellites. *IEEE Transactions on Geoscience and Remote Sensing* 51, 7 (2013), 3826–3852. https://doi.org/10.1109/TGRS.2012.2227333

[3] Daoyu Lin, Guangluan Xu, Xiaoke Wang, Yang Wang, Xian Sun, and Kun Fu. 2019. A Remote Sensing Image Dataset for Cloud Removal. *CoRR* abs/1901.00600 (2019). arXiv:1901.00600 http://arxiv.org/abs/1901.00600

[4] Guilin Liu, Fitsum A. Reda, Kevin J. Shih, Ting-Chun Wang, Andrew Tao, and Bryan Catanzaro. 2018. Image Inpainting for Irregular Holes Using Partial Convolutions. *CoRR* abs/1804.07723 (2018). arXiv:1804.07723 http://arxiv.org/abs/1804.07723

[5] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. 2015. Deep Learning Face Attributes in the Wild. In *Proceedings of International Conference on Computer Vision (ICCV)*.

[6] Sorour Mohajerani, Thomas A. Krammer, and Parvaneh Saeedi. 2018. A Cloud Detection Algorithm for Remote Sensing Images Using Fully Convolutional Neural Networks. In *20th IEEE International Workshop on Multimedia Signal Processing, MMSP 2018, Vancouver, BC, Canada, August 29-31, 2018*. IEEE, 1–5. https://doi.org/10.1109/MMSP.2018.8547095

[7] Sorour Mohajerani and Parvaneh Saeedi. 2019. Cloud-Net: An End-To-End Cloud Detection Algorithm for Landsat 8 Imagery. In *2019 IEEE International Geoscience and Remote Sensing Symposium, IGARSS 2019, Yokohama, Japan, July 28 - August 2, 2019*. IEEE, 1029–1032. https://doi.org/10.1109/IGARSS.2019.8898776

[8] EO Research. 2020. Cloud Masks at Your Service: State-of-the-art cloud masks now available on Sentinel Hub. https://medium.com/sentinel-hub/cloud-masks-at-your-service-6e5b2cb2ce8a

[9] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2015 - 18th International Conference Munich, Germany, October 5 - 9, 2015, Proceedings, Part III (Lecture Notes in Computer Science, Vol. 9351)*, Nassir Navab, Joachim Hornegger, William M. Wells III, and Alejandro F. Frangi (Eds.). Springer, 234–241. https://doi.org/10.1007/978-3-319-24574-4_28

[10] Vishnu Sarukkai, Anirudh Jain, Burak Uzkent, and Stefano Ermon. 2020. Cloud Removal in Satellite Images Using Spatiotemporal Generative Networks. In *IEEE Winter Conference on Applications of Computer Vision, WACV 2020, Snowmass Village, CO, USA, March 1-5, 2020*. IEEE, 1785–1794. https://doi.org/10.1109/WACV45572.2020.9093564

[11] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. 2004. Image quality assessment: from error visibility to structural similarity. *IEEE Trans. Image Process.* 13, 4 (2004), 600–612. https://doi.org/10.1109/TIP.2003.819861

[12] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S. Huang. 2019. Free-Form Image Inpainting With Gated Convolution. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*. IEEE, 4470–4479. https://doi.org/10.1109/ICCV.2019.00457

[13] Bolei Zhou, Àgata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. 2018. Places: A 10 Million Image Database for Scene Recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* 40, 6 (2018), 1452–1464. https://doi.org/10.1109/TPAMI.2017.2723009