Data Challenge 6: Cloud identification & Noise Removal & Image Compression

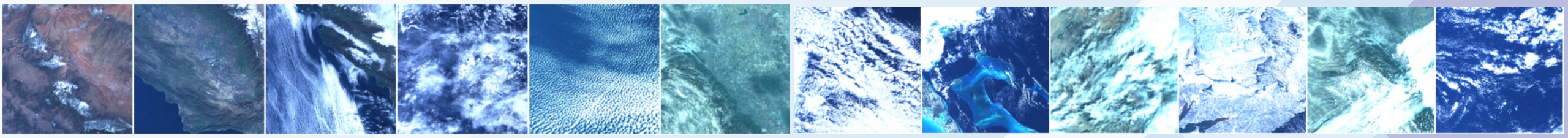
Annie Wei
Wilson Wong
Julian Halloy
Nicole Chow
Bella Zhao

Introduction

Satellite data is important for many environmental sciences, as they give scientists a bird's eye view of large areas of the Earth. Modern orbital sensor allow them to collect additional research input remotely with high precision in order to employ data-intensive analysis workflows. For example, the Sentinel-3 satellite collects images that are used by scientists for a wide variety of research tasks, such as monitoring ocean and land surface temperatures, bootstrapping models for weather forecasts or atmospheric conditions' prediction, and modeling and predicting climate change.

Sentinel-3 Satellite:

A dataset consisting of 1024×1024 red-green-blue (RGB) images generated from the Sentinel-3 satellite via the Ocean and Land Color Instrument (OLCI). We downloaded a smcefr-mini dataset which consists of 880 selected images from Sentinel-3 Satellite as the object of our study.



A sample of 12 images from the smcefr-mini dataset

Our Challenge

(#1) Cloud Identification
What methods can be used to **segment and identify the regions of the images that are partially or completely obstructed by the cloud cover**?

(#2) Noise Removal
Consider a case where the sensor data is incomplete, of varied quality, or totally degraded. **Is there any way to take partially damaged/noisy data, and reconstruct something "closer" to the original sensor reading?** Be sure to consider the measure "closeness" carefully. For example, can useful metrics (PSNR, MSE) be used to compare performance of various methods? For this task, we suggest creating copies of the "ground truth" dataset, and then adding a random amount of noise, followed by processing the copy as input.

(#3) Image Compression
To preserve data integrity, the dataset is provided as PNG, in the lossless data compression format mode. However, for a variety of purposes, it would be useful to allow a small amount of error in exchange for an appreciable size reduction. **What can research methods be used to save space while keeping the best quality possible?**

CONTENTES

#01 Cloud Identification

— K-Means Clustering
— Automatic Thresholding

#02 Image Compression

— FFT
— Wavelet
— GAN
— HiFiC

Cloud Identification: K-Means clustering

Background

K-Means clustering is an unsupervised learning algorithm that can be used to segment data into K clusters. It is particularly useful for identifying distinct differences in images based on pixel red-green-blue (RGB) values. This algorithm should prove applicable to our data since the clouds are white and typically in high contrast with the blue ocean and green land.

The K-Means clustering algorithm works by initializing K random centroids and assigning the data points to one of the centroids. The centroids are then adjusted based on the minimization of the distances of all associated data points. The function to be minimized is as follows:

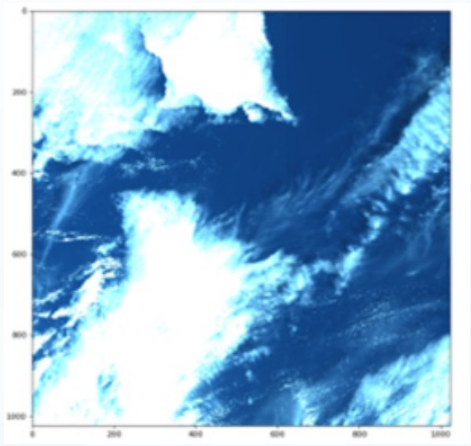
$$\sum_{i=1}^{m}\sum_{k=1}^{K} w_{ik}||x^i - \mu_k||^2 \qquad (1)$$

where w_{ik} is 1 when data point x^i is part of cluster k, and 0 otherwise. μ_k is the centroid of cluster k. The equation is minimized with respect to w_{ik} and μ_k separately. This adjusts the cluster assignments and then the centroids, respectively.

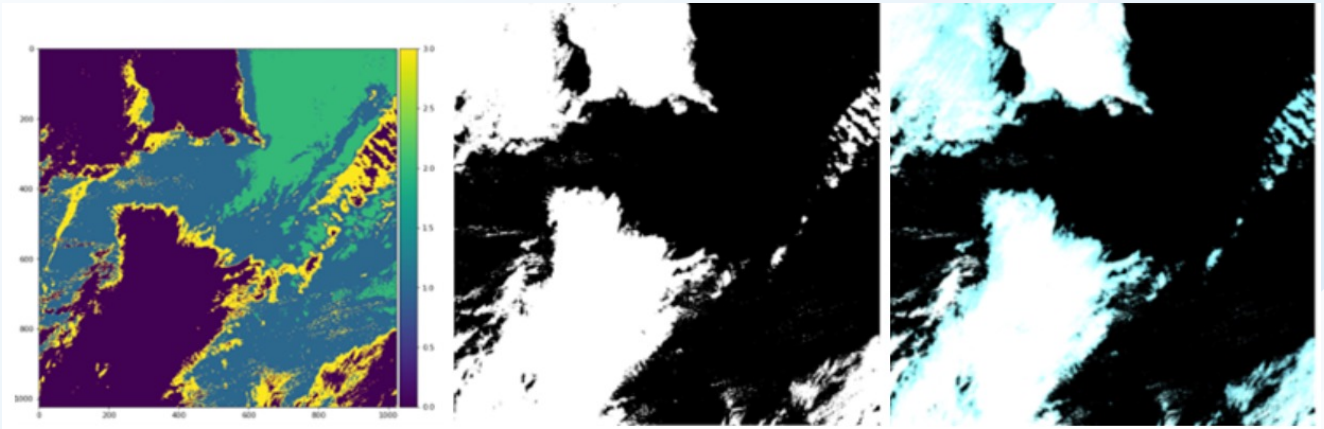
Cloud Identification: K-Means clustering

Results

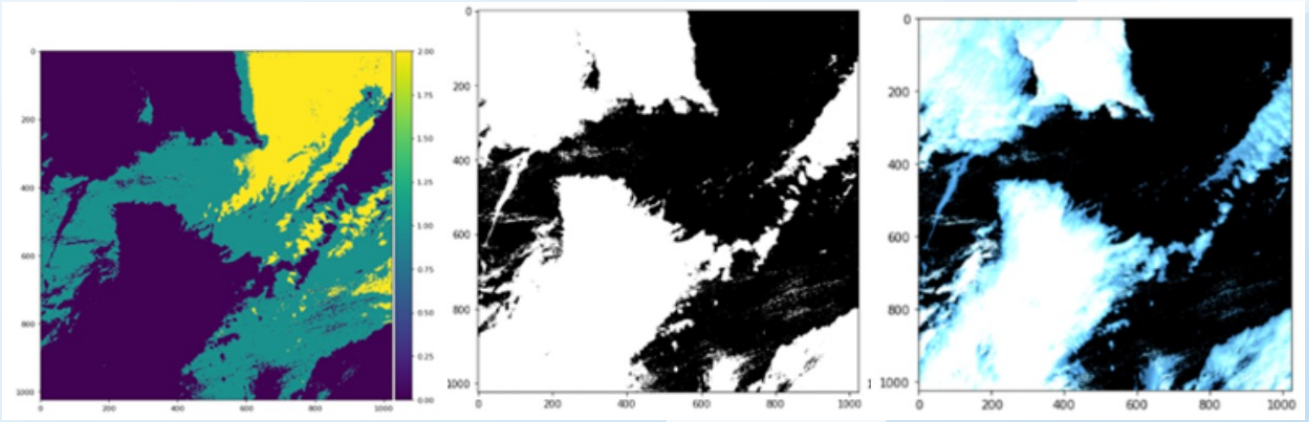
Using the K-Means function of the scikit-learn python module, we performed K-Means clustering on the image for both four and three clusters.
The results for 4 clusters were as follows:



Original Image



Next, we try reducing the number of segments to three:



Cloud Identification: K-Means clustering

Limitations

1. Areas with snow are hard to differentiate between clouds due to both being the same color.
2. The K-Means clustering algorithm does not have labels for the clusters, so it is necessary for a human to manually identify a cluster as clouds. This is undesirable since it is time-consuming.

Cloud Identification: Automatic Thresholding

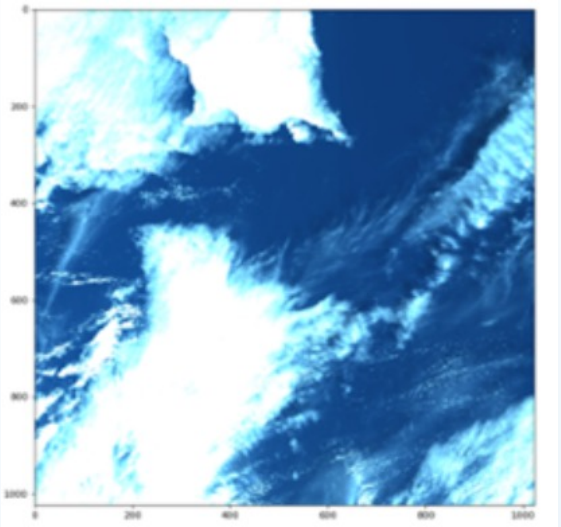
Background

Thresholding is simply the selection of certain pixels based on a set threshold t. Images are typically converted to grayscale and normalized from 0 to 1. Each value in the image is compared to t and we create a binary mask based on it.

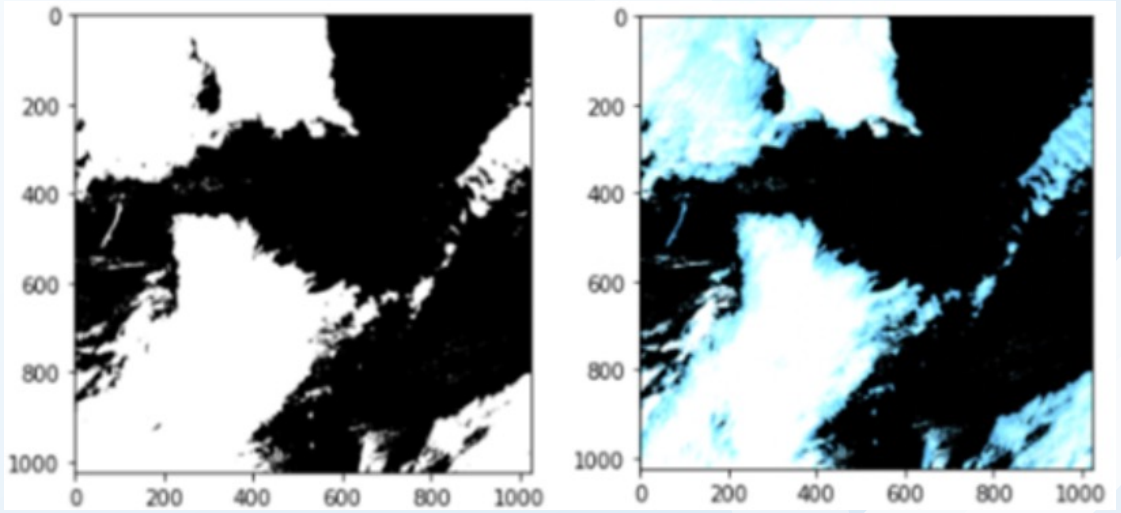
This can be effective in segmenting an image, but usually requires the use of a histogram to correctly set t. Instead, we can make use of automatic thresholding to set t for us. The scikit-image module in python has a function that finds the threshold t using Otsu's method. This method essentially finds a value between peaks in a grayscale histogram.

Cloud Identification: Automatic Thresholding

Results & Limitations



Original image



Results from automatic thresholding

1. This method reduces the data from RGB to grayscale. This loss in data complexity may decrease the accuracy of image segmentation.
2. The binary classification may be effective for images that are simply clouds and ocean or clouds and land, but images with all three sections may not work correctly.

Image Compression

Introduction

- Goal: For a given image, minimize the length of the sequence of bits needed to represent it, while preserving information of acceptable quality.
- Performance: Compression Ratio (CR), Bit-Per-Pixel Ratio (BPP), Mean Square Error (MSE), Peak signal-to-noise Ratio (PSNR).
  - CR: the compressed image is stored using CR% of the initial storage size.
  - BPP: the number of bits used to store one pixel of the image.
    grayscale image: initial BPP=8; true-color image: initial BPP=24.
  - MSE= $\frac{1}{m\,n}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i,j)-K(i,j)]^2$
  - PSNR= $10\cdot\log_{10}\left(\frac{MAX_I^2}{MSE}\right) = 20\cdot\log_{10}\left(\frac{MAX_I}{\sqrt{MSE}}\right) = 20\cdot\log_{10}(MAX_I) - 10\cdot\log_{10}(MSE).$

Image Compression: FFT

Backgrounds

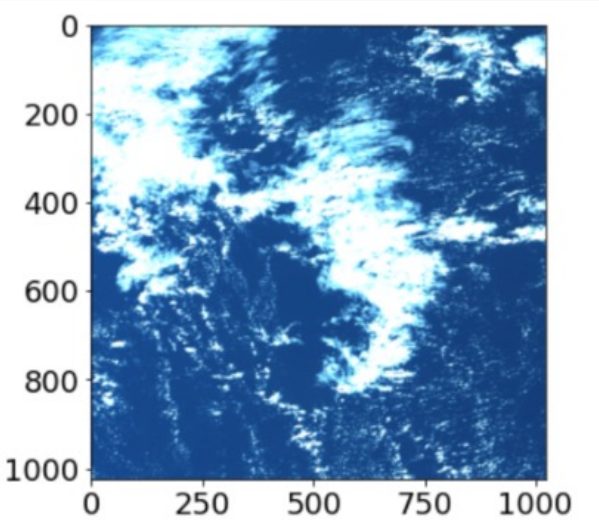
Fourier analysis is fundamentally a method for expressing a function as a sum of periodic components, and for recovering the function from those components. In two dimensions, the discrete Fourier transform (DFT) is defined as:

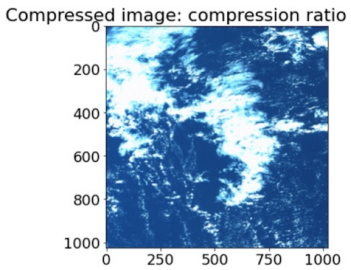
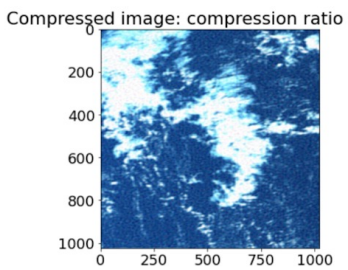
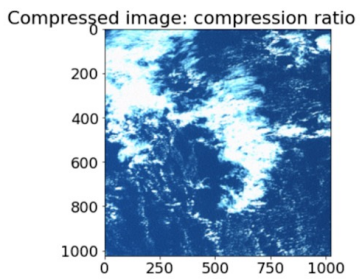
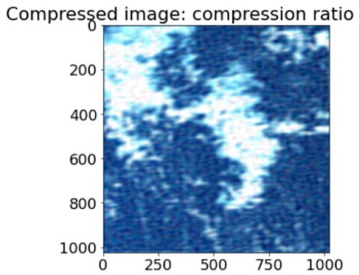
$$A_{kl} = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} a_{mn} \exp\left\{-2\pi i \left(\frac{mk}{M} + \frac{nl}{N}\right)\right\} \qquad (2)$$

Image Compression: FFT

Results & Limitations

2D-FFT works for two-dimensional images, so we first split the three channels of our RGB images and apply FFT to the R, G, and B channels, respectively, and then merge the results to form the reconstructed images. To realize compression, during FFT, we keep the largest 10%, 5%, 1%, and 0.2% Fourier coefficients and truncate other Fourier coefficients that are smaller than the threshold. The results for compression with 4 different compression ratios were as follows:



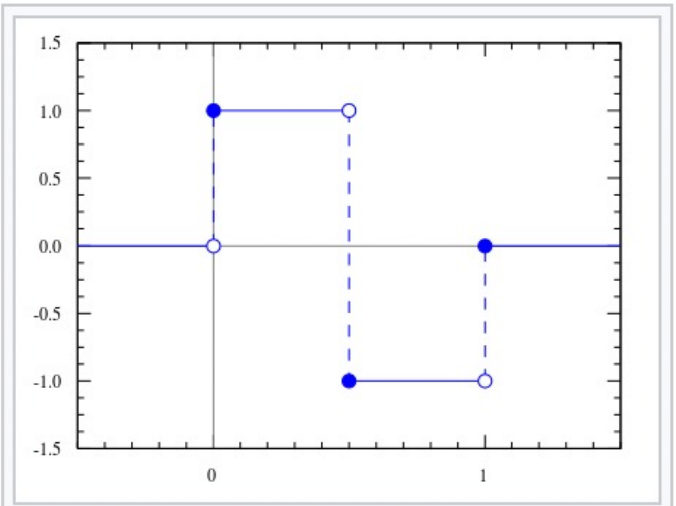
	MSE: 145.2875 PSNR: 26.5085 SSIM1: 0.5402851		MSE: 612.7738 PSNR: 20.2578 SSIM: 0.38167763
	MSE: 257.1953 PSNR: 24.0282 SSIM: 0.4774895		MSE: 1035.3549 PSNR: 17.9799 SSIM: 0.38824356

FFT can only work on one 2-dimensional matrix at a time, so when compressing RGB images, it will take a longer time than other methods.

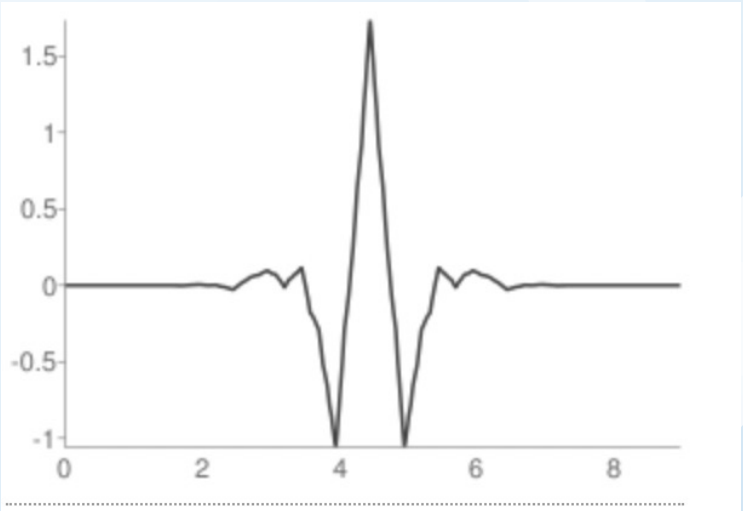
Image Compression: Wavelet

Backgrounds

Wavelet compression is a form of data compression well suited for image compression, and it begins with a wavelet transform that produces a coefficient for each pixel (there is no compression yet since it is only a transform). Because the information is statistically concentrated in just a few coefficients, a few of these coefficients contain the majority of the data. These are used to encode the data.



The Haar wavelet



Bior 4.4 wavelet

Image Compression: Wavelet

Results

2D wavelet compression was performed using both the command line and the built-in Wavelet Analyzer App in MATLAB. Wavelet compression was effective in bringing down the bytes per pixel (BPP) significantly from 24 to 0.71. The MSE and PSNR are still acceptable values, and it is difficult to spot any differences in the image. The results are better than the FFT method and less time-consuming.

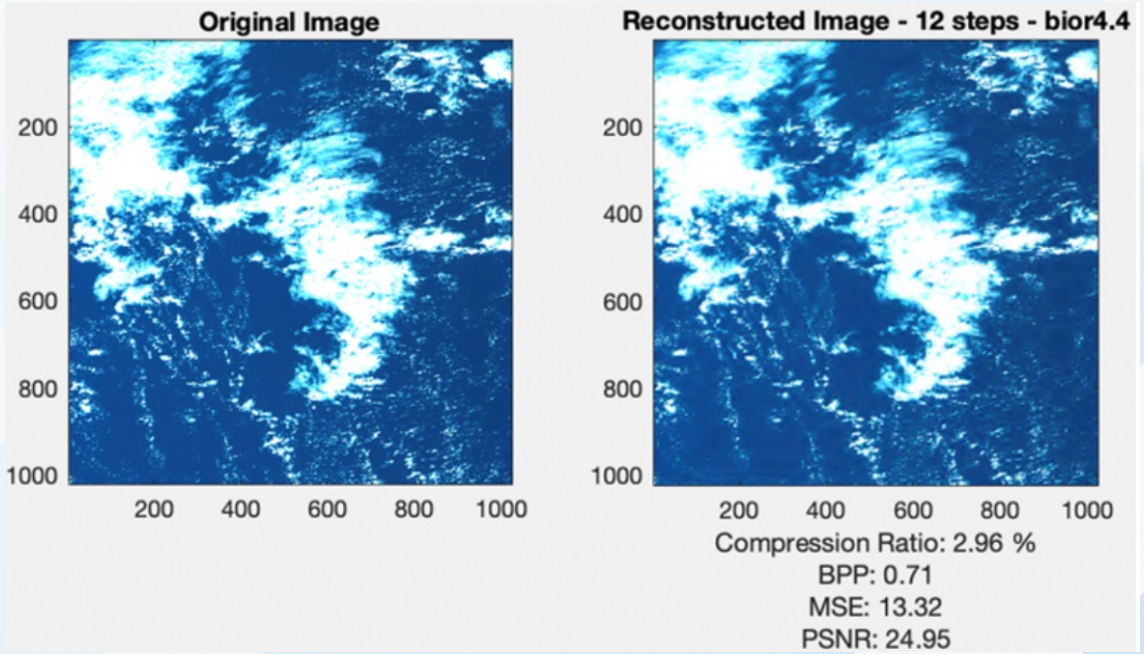


Image Compression: GAN

Background

Using Encoder-Decoder models based on deep neural networks trained with L1 and L2 loss is a popular way to do image compression. The problem with this approach is that the reconstruction of the image from compressed format is not realistic, miss a lot of detail and is often blurry. We can address this problem using a novel approach based on GAN.

A Generative Adversarial Network (GAN) consists of two networks called the generator and the discriminator. During training, the network tries to minimize an adversarial loss function. To achieve this the generator tries to create images that cannot be differentiated from true images while the discriminator tries to correctly classify images as real or generated. The discriminator is constructed just for the training purposes and is discarded after training.

Image Compression: GAN

Structure

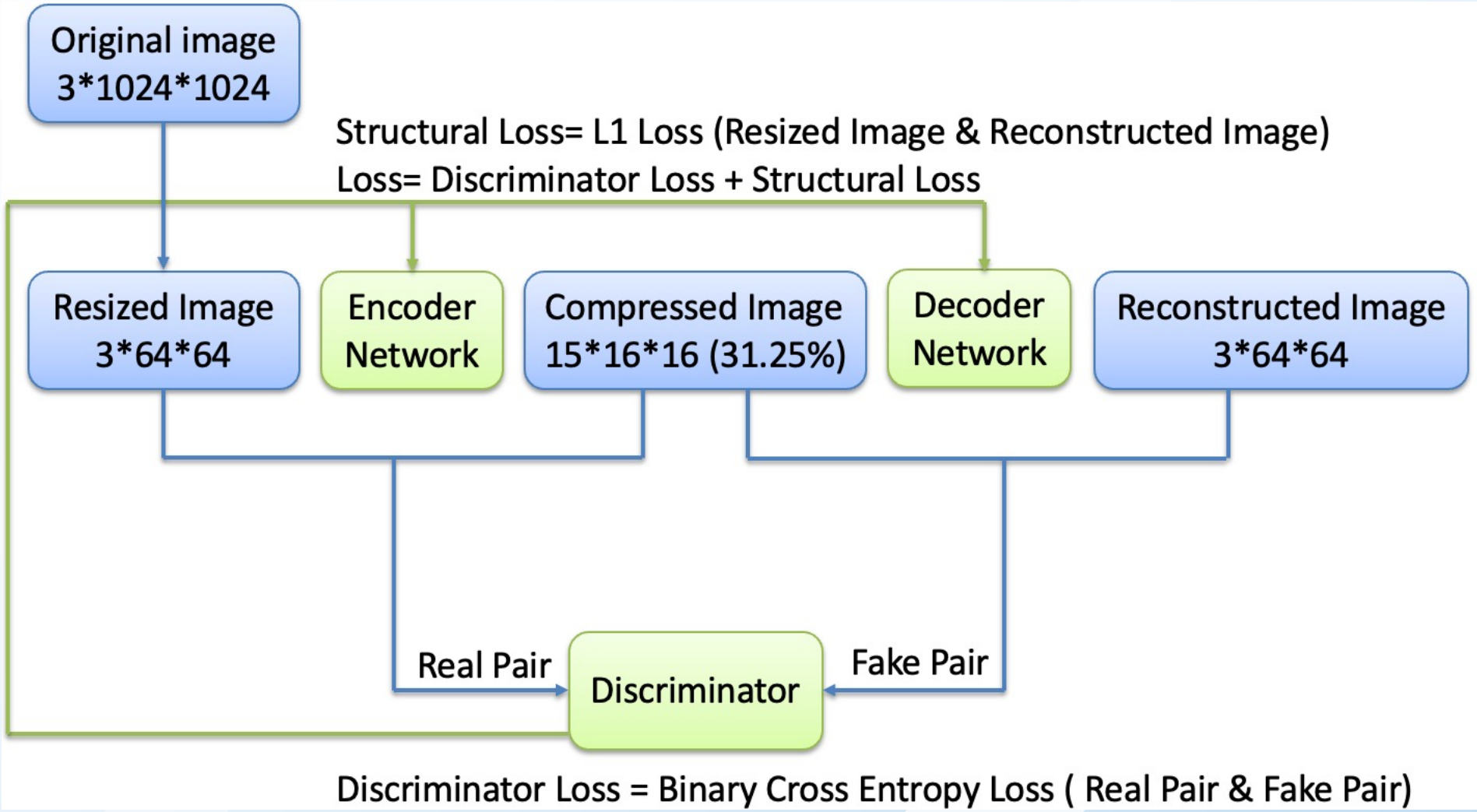
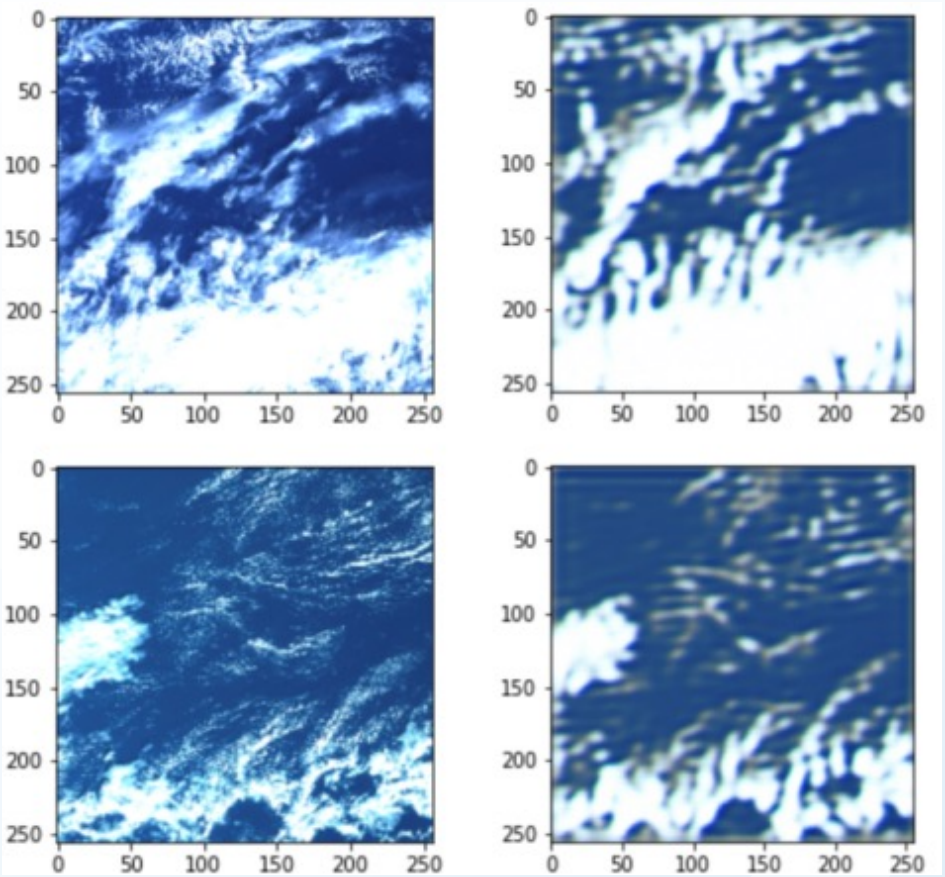


Image Compression: GAN

Results & Limitations



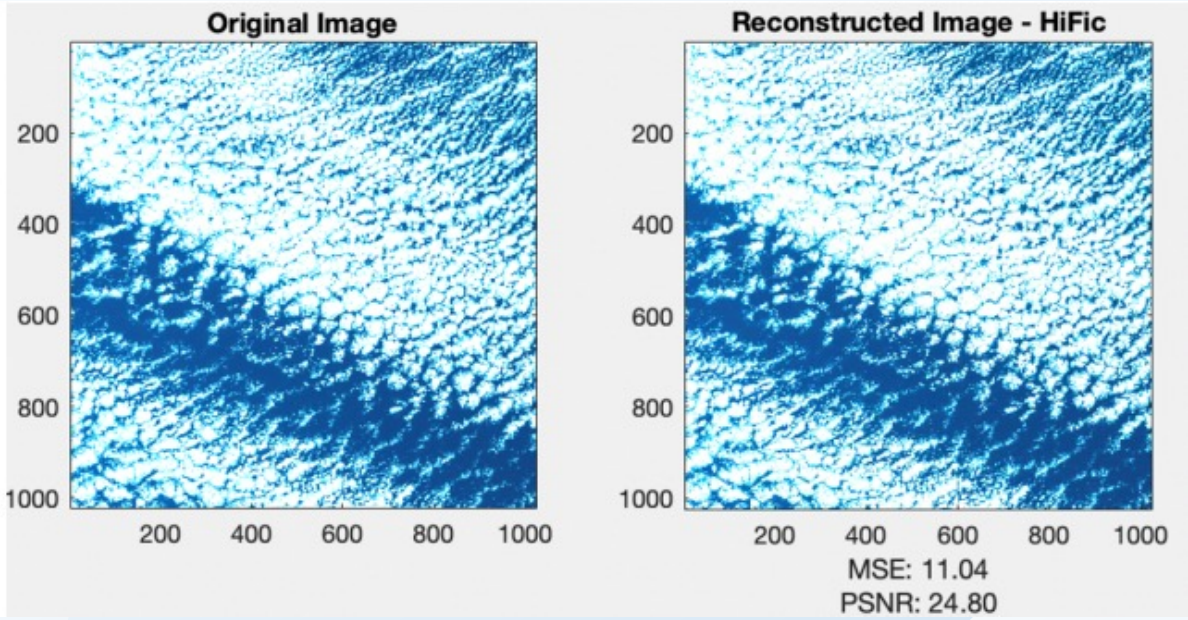
The results for GAN

1. Training such GAN model is time-consuming and requires a large amount of training data as input. Currently, we have only trained 20 epochs for 880 images, and the results are not very satisfactory.
2. Because the satellite images are different from the usual images, the graininess is stronger and the colors of the images are jumpier, so it is also more difficult to train the neural network.

Image Compression: HiFiC

Background & Results

HiFiC is a GAN-based image compression tool, which has been trained to perform image compression about four times better than JPEG. The pre-trained models are freely available to download or open via Google Colab directly. The HiFiC model can achieve 0.237 BPP. In our testing we achieved a BPP of 0.5798 using the HiFiC example on Google Colab.



The reconstructed image's MSE and PSNR are only slightly worse than the wavelet compression while saving significantly more space. The reconstructed image shows virtually no visual difference from the original.

Q&A