Cloud Segmentation, Denoising, and Compression Techniques for use on Sentinel-3 Satellite Data

Ming Chi Wong[1], Yuqiao Zhao[1] Yulin Wei[2], Tsz Ching Chow[2], and Julian Halloy[3]

[1] City University of Hong Kong
[2] The Chinese University of Hong Kong
[3] The University of Tennessee, Knoxville

Abstract. Satellite images of the Earth's surface have a multitude of uses. However, this data comes with some inherent issues. One such issue is cloud coverage. We propose the use of image segmentation to identify clouds in images. Another problem is the size of the data, which must be transferred over the satellite's limited connection bandwidth. We propose several compression methods for reducing data size effectively. The data is of varying quality, so the use of noise removal techniques can improve the accuracy and usability of the data.

Keywords: Deep Learning, GANs, Super-resolution, Image Compression

1 Problem 1: Cloud Identification

We propose the use of K-Means clustering and automatic thresholding for identifying the portions of the image that have cloud cover.

1.1 K-Means Clustering

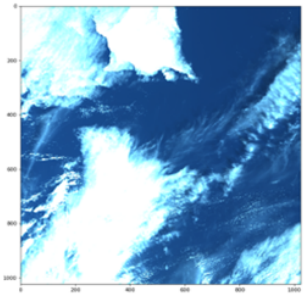
K-Means clustering is an unsupervised learning algorithm that can be used to segment data into K clusters. It is particularly useful for identifying distinct differences in images based on pixel red-green-blue (RGB) values [1]. This algorithm should prove applicable to our data since the clouds are white and typically in high contrast with the blue ocean and green land.

Background 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:

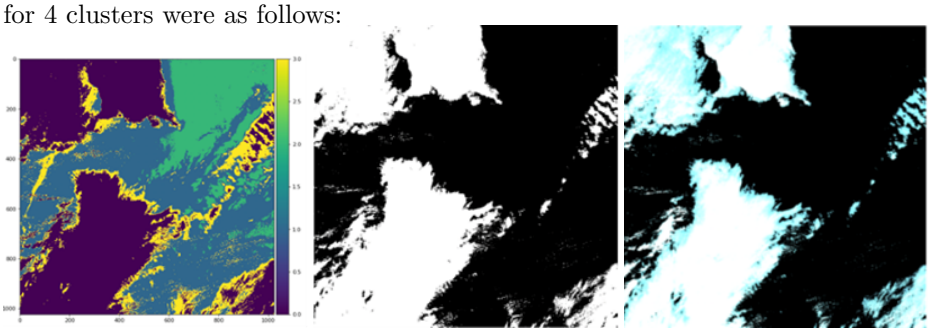
$$\sum_{i=1}^{m}\sum_{k=1}^{K} w_{ik}||x^i - \mu_k||^2 \tag{1}$$

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 [2].

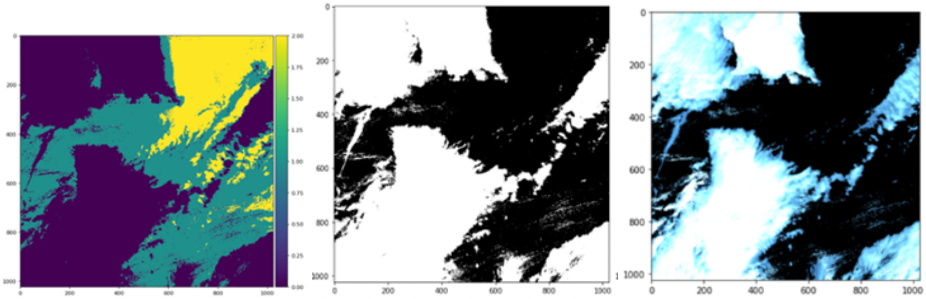
Results We selected the following image for an example:



Using the KMeans 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:



The segmentation appears to cluster the clouds correctly, but four clusters appear to be too many as it is segmenting both the ocean and clouds in two. Next, we try reducing the number of segments to three:



This works better as the clouds are kept as one cluster. The ocean is still in two clusters but this is not relevant to cloud identification. Also, with three segments we have a cluster for each image section: land, oceans, and clouds. We believe this makes three segments the most effective in K-Means clustering.

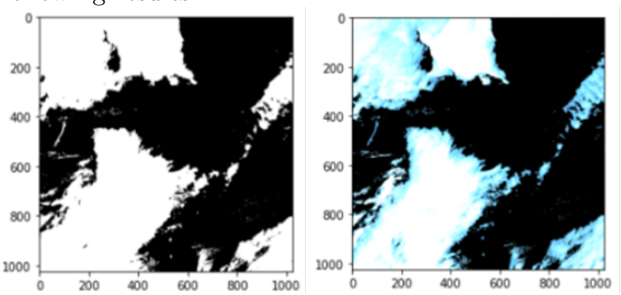
Limitations While K-Means clustering is effective for much of the dataset, there are some limitations. Areas with snow are hard to differentiate between clouds due to both being the same color. The K-Means clustering algorithm also does not have labels for the clusters, so it is necessary for a human to manually identify a cluster as clouds. This is undesirable since it is time-consuming.

1.2 Automatic Thresholding

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 [3].

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 [3].

Results We applied automatic thresholding to the same image and received the following results:



The image is segmented correctly into clouds and the ocean. The method is effective for a portion of the dataset.

Limitations This method reduces the data from RGB to grayscale. This loss in data complexity may decrease the accuracy of image segmentation. Also, 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.

1.3 Suggested Future Approaches

Some of the limitations of these methods can be overcome with the use of supervised learning. With labeled data, we can train a model that can specifically identify and segment clouds. However, this will require the creation of labeled data, which takes a significant amount of time. The U-Net architecture [14] has proven to be quite useful in performing image segmentation. Part of our team

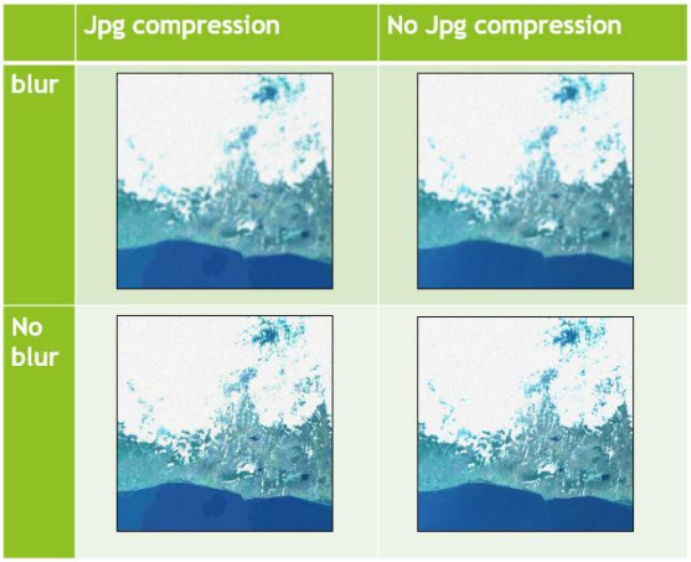
is working on implementing U-Net in MagmaDNN, a neural network framework developed at the University of Tennessee [15]. It is our ideal goal to implement U-Net with MagmaDNN for performing cloud segmentation since MagmaDNN will provide the speed to train the dataset efficiently.

2 Problem 2: Noise Removal

The satellite data can be of varied quality and can contain noise and degraded data. This calls for a means to reconstruct data to a state close to the ground truth.

2.1 Data Augmentation

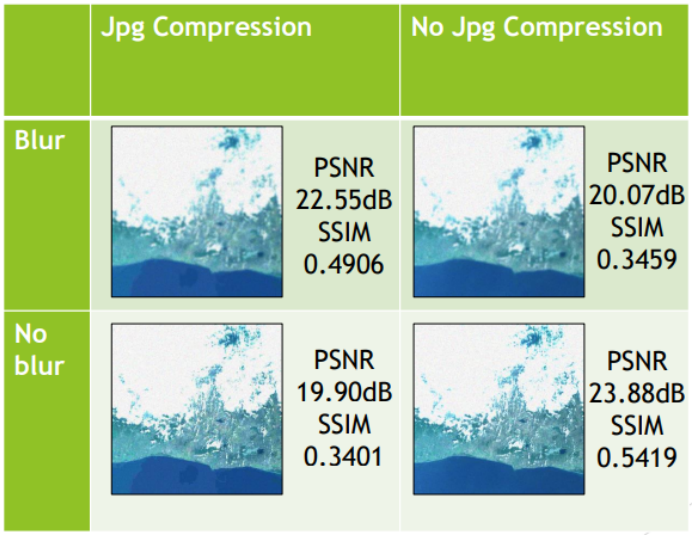
To create the degraded data, we used a combination of jpg compression and Gaussian blur to downsize the image to 50% of its original size. The following shows an image downsized using the different methods:



Noise can then be added using the python module NumPy. This is done with NumPy's normal distribution function with mean 0 and standard deviation 0.1. The values are then added to the image to create noise.

2.2 Bicubic Resize

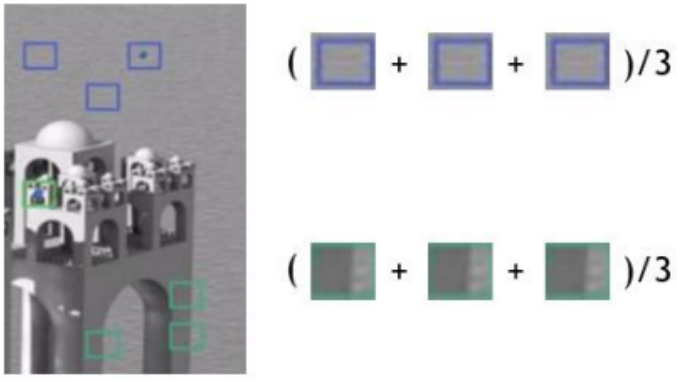
Bicubic resizing is a method that fits cubic functions to the two-dimensional image and uses those to estimate new pixel values during upscaling. We used this primarily as a control group to compare the other results. The following is the result:



The peak signal-to-noise ratio (PSNR) is a measure of the noise between the original and upscaled image. The higher the PSNR, the less noise there is. SSIM is a measure that focuses on the structural information of the original and reconstructed images. The SSIM value is between 0 and 1, with 1 being the most similar.

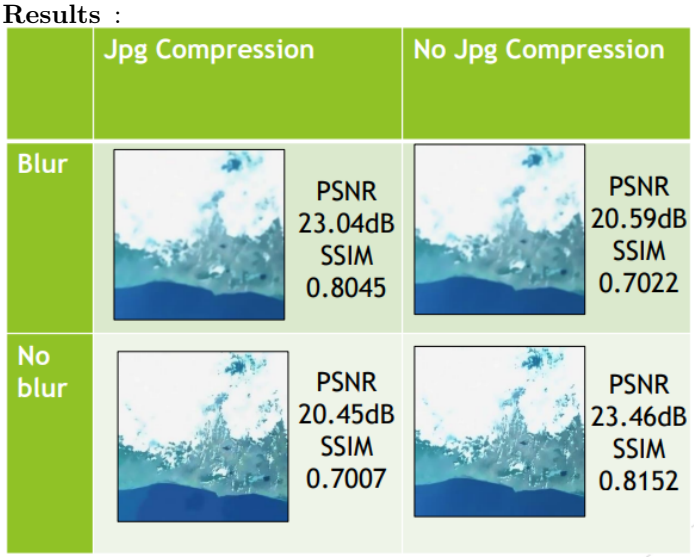
2.3 Non-local Means Denoising

Background The non-local means denoising algorithm is available in the OpenCV python module. This finds common patterns in images and averages them to remove noise.



We use the OpenCV fastNlMeansDenoisingColored() function for this [4].

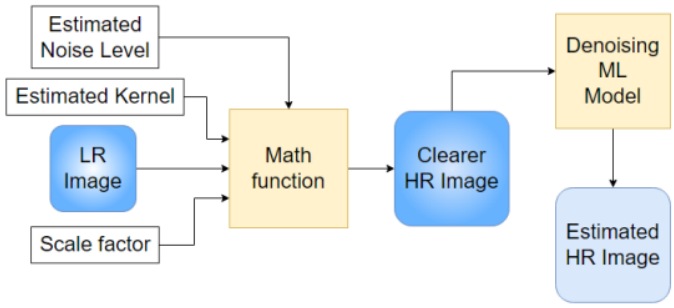
Results :



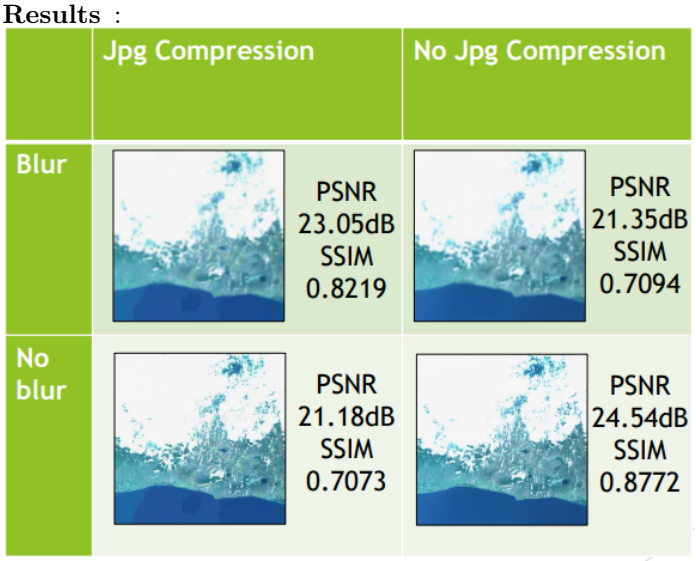
Non-local means denoising achieves an average SSIM of 0.7557, much better than the average of bicubic resizing (average SSIM of 0.4296). The algorithm is slower than bicubic resizing by about 7 times.

2.4 USRNet

Background Unfolding super-resolution network [7] (USRNet) is a neural network that takes four inputs: the low resolution (LR) image, estimated kernel, estimated noise level, and scale factor. Using these it creates a high-resolution image and then denoises that to create its predicted image [5].



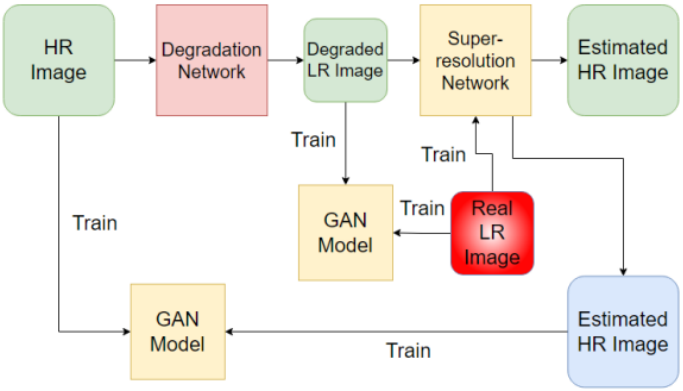
Results :



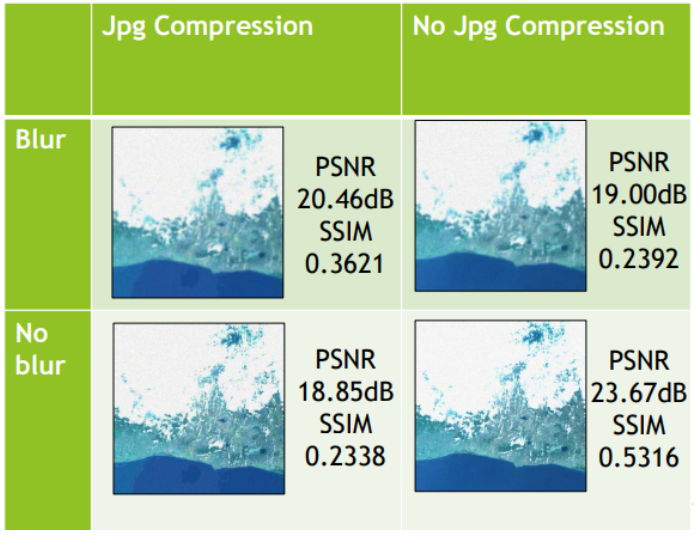
The USRNet achieves an even higher average SSIM of 0.7790. This comes at the cost of time since it takes about 1.6 times the time of the non-local means denoising algorithm.

2.5 DASR

Background Distance aware super-resolution [8] (DASR) is a neural network that takes one input: the low-resolution image. It also contains a degradation network that creates its own degraded low-resolution image [6].



Results :



DASR only achieves an average SSIM of 0.3417. It is also slower than USRNet, taking about 3 times longer. The model is a general model which we think could be improved if we trained it for our specific dataset. However, we did not have the time to do this.

2.6 Suggested Future Approaches

We would like to run these methods on an edge device, such as the Nvidia Jetson Nano [16]. The Jetson Nano has a graphics processing unit (GPU) which allows for the acceleration of the USRNet and DASR models. The Jetson Nano is a small form factor that allows for a portable processing unit. A unit, such as the Nano, could be made a part of the satellite, and perform denoising locally.

3 Compression

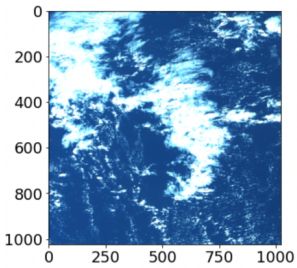
Image data can be quite large and the Sentinel-3 Satellite has limited bandwidth for transferring data. It is logical, therefore, to compress images in order to save space and transfer time. The use of lossy data compression can provide a significant decrease in data size with a small error. We propose the use of fast Fourier transform (FFT), wavelet, generative adversarial networks (GANs), and high-fidelity generative image compression (HiFiC) as compression methods.

3.1 FFT

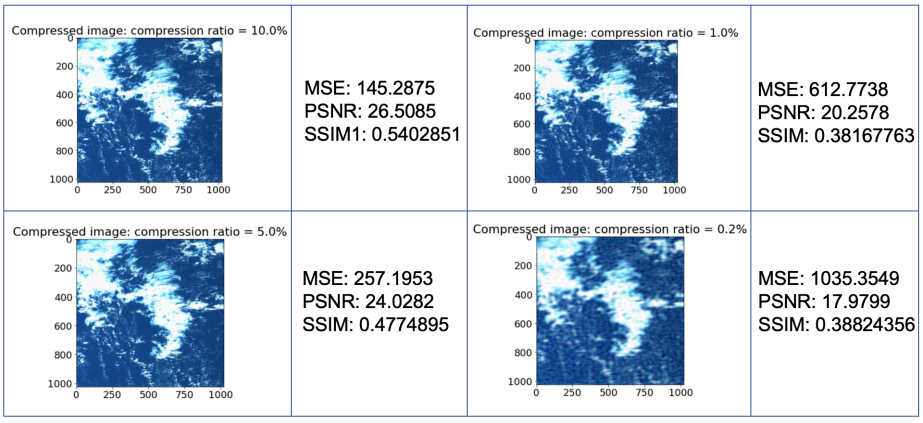
Background Fourier analysis is fundamentally a method for expressing a function as a sum of periodic components, and for recovering the function from those components[17]. In two dimensions, the discrete Fourier transform (DFT) is defined as:

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

Results We selected the following image for an example:



2D-FFT works for two-dimensional images, that is, grayscale 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, so we can get images with a compression ratio of 10%, 5%, 1%, 0.2% respectively. The results for compression with 4 different compression ratios were as follows:



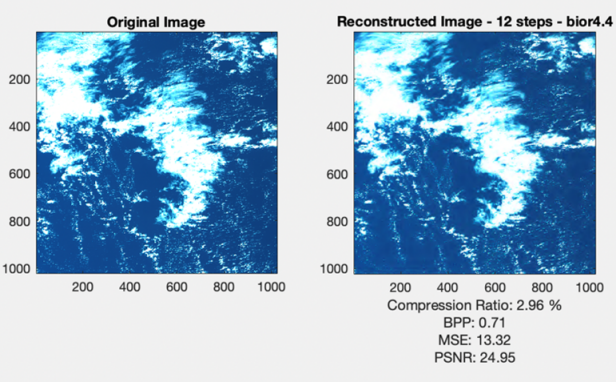
This method can get good performance with compression ratios equal to 10% and 5%, but with smaller compression ratios, the mean squared error gets much larger, and the images become much more blurry.

Limitations 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.

3.2 Wavelet

Background Wavelet compression begins with a wavelet transform that produces a coefficient for each pixel. A few of these coefficients contain the majority of the data. These are used to encode the data.

Results 2D wavelet compression was performed using both the command line and the built-in Wavelet Analyzer App in MATLAB [11].

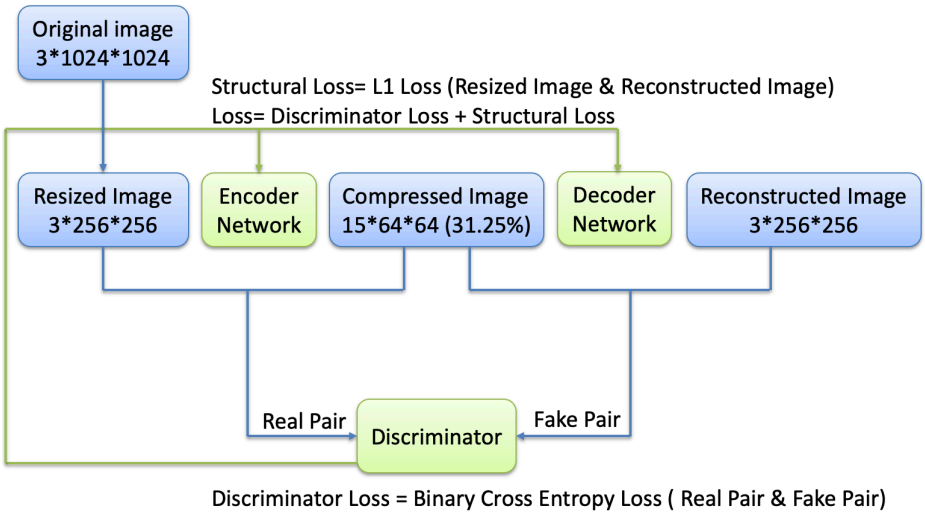


Wavelet compression was effective in bringing down the bytes per pixel (BPP) significantly from 24 to 0.71. The MSE and PSNR are still acceptable values and it is difficult to spot any differences in the image. The results are better than the FFT method and less time-consuming.

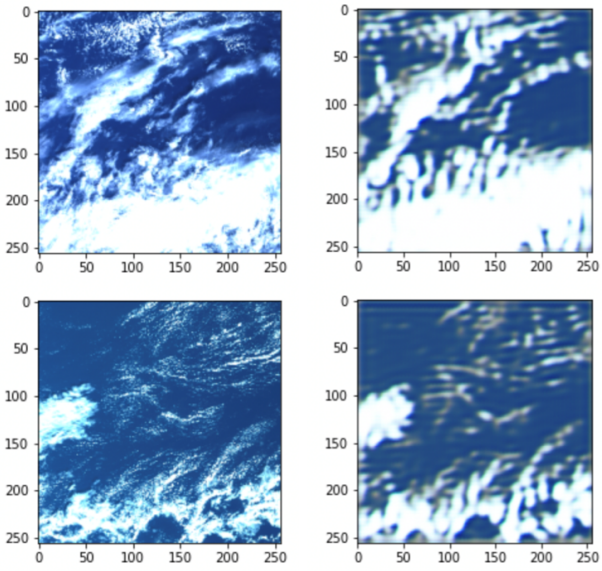
3.3 GAN

Background Generative adversarial networks contain a generator and discriminator. The generator generates the image, and the discriminator attempts to classify it as real or generated. The generator is trying to create images that the discriminator cannot tell are generated[9]. This type of network can be used with an Encoder-Decoder model to create compressed images that are similar

to the original [16].



Results The images produced by the GAN represented the original image but looked somewhat blurry.

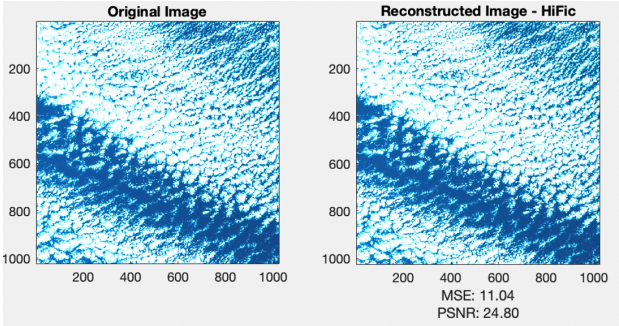


Limitations Training such GAN model is time-consuming and requires a large amount of training data as input. Currently, we have only trained 20 epochs for 880 images, and the results are not very satisfactory. At the same time, because the satellite images are different from the usual images, the graininess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

3.4 HiFiC

Background 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 [12].

Results In our testing we achieved a BPP of 0.5798 using the HiFiC example on Google Colab [13].



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.

3.5 Conclusion

So far, among the four methods, we have tried, FFT, wavelet, GAN, and HiFiC, both wavelet and HiFiC have achieved good results. For GAN, we also believe it can achieve better image compression in the subsequent improvement process.

3.6 Suggested Future Approaches

In the future, we would custom train the HiFiC for our dataset so we can achieve better compression. These networks could also be run on a Jetson Nano. This could allow for a portable compression device that could be located where it is most needed. If possible, compression could even be performed on the satellite to reduce bandwidth needs.

References

1. Edeza, T.: Image Processing with Python - Unsupervised Learning for Image Segmentation. Medium, Towards Data Science, (2021). https://towardsdatascience.com/image-processing-with-python-unsupervised-learning-for-image-segmentation-90ebd23d91a4

2. Dabbura, I.: K-means Clustering: Algorithm, Applications, Evaluation Methods, and Drawbacks. Medium, Towards Data Science, (2018). https://towardsdatascience.com/k-means-clustering-algorithm-applications-evaluation-methods-and-drawbacks-aa03e644b48a
3. Image Processing with Python: Thresholding https://datacarpentry.org/image-processing/07-thresholding/
4. Image Denoising https://docs.opencv.org/3.4/d5/d69/tutorial_py_non_local_means.html
5. Zhang, K., Gool, L., Timofte, R.: Deep Unfolding Network for Image Super-Resolution. CVPR (2020) pp. 3214-3223. https://doi.org/10.48550/arXiv.2003.10428
6. Kim, S., Sim, H., Kim, M.: KOALAnet: Blind Super-Resolution using Kernel-Oriented Adaptive Local Adjustment. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2021). pp. 10611-10620. https://doi.org/10.48550/arXiv.2012.08103
7. Deep Unfolding Network for Image Super-Resolution https://github.com/cszn/USRNet
8. Distance Aware Super-Resolution (DASR). https://github.com/ShuhangGu/DASR
9. Tolunay, E., Ghalayini, A.: Generative Neural Network Based Image Compression. (2018). https://cs229.stanford.edu/proj2018/report/44.pdf
10. Image Compression Using GAN. https://github.com/crbanala/image_compression
11. Wavelet Analyzer App. https://www.mathworks.com/help/wavelet/ref/waveletanalyzer-app.html
12. Mentzer, F., Toderici, G., Tschannen, M., Agustsson, E.: High-Fidelity Generative Image Compression. (2020). https://hific.github.io/
13. HiFiC Colab https://colab.research.google.com/github/tensorflow/compression/blob/master/models/hific/colab.ipynb
14. Ronneberger, O., Fischer, P., Brox, T.: U-Net: Convolutional Networks for Biomedical Image Segmentation (2015). https://arxiv.org/pdf/1505.04597.pdf
15. Nichols, D., Wong, K., Tomov, S., Ng, L. Sihan, C., Gessinger, A.: MagmaDNN: Accelerated Deep Learning Using MAGMA (2019). https://www.icl.utk.edu/files/publications/2019/icl-utk-1189-2019.pdf
16. Nvidia Jetson Nano Developer Kit. https://developer.nvidia.com/embedded/jetson-nano-developer-kit
17. Discrete Fourier Transform. https://NumPy.org/doc/stable/reference/routines.fft.html